# Exploiting Semantic Web Knowledge Graphs in Data Mining


Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

presented by
Petar Ristoski

Mannheim, 2017

Dekan: Dr. Bernd Lübcke, Universität Mannheim
Referent: Professor Dr. Heiko Paulheim, Universität Mannheim
Korreferent: Professor Dr. Simone Paolo Ponzetto, Universität Mannheim

Tag der mündlichen Prüfung: 15 Januar 2018

# Abstract

Data Mining and Knowledge Discovery in Databases (KDD) is a research field concerned with deriving higher-level insights from data. The tasks performed in that field are knowledge intensive and can often benefit from using additional knowledge from various sources. Therefore, many approaches have been proposed in this area that combine Semantic Web data with the data mining and knowledge discovery process. Semantic Web knowledge graphs are a backbone of many information systems that require access to structured knowledge. Such knowledge graphs contain factual knowledge about real word entities and the relations between them, which can be utilized in various natural language processing, information retrieval, and any data mining applications. Following the principles of the Semantic Web, Semantic Web knowledge graphs are publicly available as Linked Open Data. Linked Open Data is an open, interlinked collection of datasets in machine-interpretable form, covering most of the real world domains.

In this thesis, we investigate the hypothesis if Semantic Web knowledge graphs can be exploited as background knowledge in different steps of the knowledge discovery process, and different data mining tasks. More precisely, we aim to show that Semantic Web knowledge graphs can be utilized for generating valuable data mining features that can be used in various data mining tasks.

Identifying, collecting and integrating useful background knowledge for a given data mining application can be a tedious and time consuming task. Furthermore, most data mining tools require features in propositional form, i.e., binary, nominal or numerical features associated with an instance, while Linked Open Data sources are usually graphs by nature. Therefore, in Part I, we evaluate unsupervised feature generation strategies from types and relations in knowledge graphs, which are used in different data mining tasks, i.e., classification, regression, and outlier detection. As the number of generated features grows rapidly with the number of instances in the dataset, we provide a strategy for feature selection in hierarchical feature space, in order to select only the most informative and most representative features for a given dataset. Furthermore, we provide an end-to-end tool for mining the Web of Linked Data, which provides functionalities for each step of the knowledge discovery process, i.e., linking local data to a Semantic Web knowledge graph, integrating features from multiple knowledge graphs, feature generation and selection, and building machine learning models. However, we show that such feature generation strategies often lead to high dimensional feature vectors even after

dimensionality reduction, and also, the reusability of such feature vectors across different datasets is limited.

In Part II, we propose an approach that circumvents the shortcomings introduced with the approaches in Part I. More precisely, we develop an approach that is able to embed complete Semantic Web knowledge graphs in a low dimensional feature space, where each entity and relation in the knowledge graph is represented as a numerical vector. Projecting such latent representations of entities into a lower dimensional feature space shows that semantically similar entities appear closer to each other. We use several Semantic Web knowledge graphs to show that such latent representation of entities have high relevance for different data mining tasks. Furthermore, we show that such features can be easily reused for different datasets and different tasks.

In Part III, we describe a list of applications that exploit Semantic Web knowledge graphs, besides the standard data mining tasks, like classification and regression. We show that the approaches developed in Part I and Part II can be used in applications in various domains. More precisely, we show that Semantic Web graphs can be exploited for analyzing statistics, building recommender systems, entity and document modeling, and taxonomy induction.

# Zusammenfassung

Data Mining und Knowledge Discovery in Databases (KDD) ist ein Forschungsbereich, der sich mit dem Extrahieren von Informationen und Fakten aus Daten beschäftigt. Aufgaben aus diesem Feld der Forschung benötigen viel Wissen und profitieren oft von Wissen aus verschiedenen Quellen. Daher wurden in diesem Bereich schon viele Ansätze vorgeschlagen, die Daten aus dem Semantic Web mit Data Mining und Knowledge Discovery Prozessen kombinieren. Semantic Web Knowledge Graphs sind dabei oft die Technologien, auf die viele Informationssysteme, welche Zugang zu strukturierten Daten benötigen, zurückgreifen. Solche Knowledge Graphs beinhalten Informationen und Fakten über Entitäten aus der realen Welt und ihre Beziehungen zueinander. Diese Informationen können von verschiedenen Natural Language Processing, Information Retrieval und Data Mining Applikationen genutzt werden. Nach dem Prinzip von Semantic Web, sind auch Semantic Web Knowledge Graphs in Form von Linked Open Data öffentlich verfügbar. Linked Open Data ist hierbei eine Sammlung von öffentlich verfügbaren Datensätzen aus verschiedenen Domänen, die miteinander verknüpft sind und in maschinenlesbarer Form vorliegen.

In dieser Dissertation beschäftigen wir uns mit der Hypothese, ob Hintergrundinformationen aus Semantic Web Knowledge Graphs sowohl in verschiedenen Schritten der Knowledge Discovery als auch in Bereichen des Data Mining genutzt werden können. Hierbei wollen wir vor allem zeigen, dass markante Data Mining Features aus Semantic Web Knowledge Graphs in einer Reihe von Data Mining Aufgaben hilfreich sind.

Das Identifizieren, Sammeln und Integrieren von nützlichen Hintergrundinformationen für eine gegebene Data Mining Anwendung ist oftmals sehr aufwendig und zeitintensiv. Zudem benötigen viele Data Mining Applikationen die Feature in Aussagenform (z.B. binäre, nominale oder nummerische Feature die zu einer Instanz gehören), wohingegen Linked Open Data Datensätze meist in Form eines Graphen vorliegt. Daher evaluieren wir in Teil I verschiedene unüberwachte Ansätze um Features aus Relationen im Knowledge Graphs zu extrahieren. Diese werden für Aufgaben aus dem Data Mining wie Klassifizierung, Regression und Erkennung von Ausreißern benutzt. Hierbei wächst die Anzahl der Feature stark mit der Anzahl der Instanzen im Datensatz, weswegen wir einen Ansatz für die Selektion von Features in einem hierarchischen Feature Space präsentieren, der die informativsten und repräsentativsten Features aus einem gegebenen Datensatz aus-

sucht. In diesem Zusammenhang stellen wir unsere End-to-end Anwendung, die Data Mining auf dem Netz von Linked Data ermöglicht, zur Verfügung. Diese unterstützt sowohl alle Schritte der Knowledge Discovery (das Vernetzen von lokalen Daten mit Daten aus dem Semantic Knowledge Graphs, die Integration von Features aus verschiedenen Graphen und die Generierung und Selektion von Features), als auch die Erzeugung von Modellen aus dem Machine Learning. Allerdings zeigen wir auch, dass unser Ansatz selbst nach der Reduktion der Feature immer noch eine sehr hohe Dimensionalität aufweist und zudem Feature Vektoren aus einem solchen Modell sich schwer auf andere Datensätze anwenden lassen.

Im zweiten Teil stellen wir einen weiteren Ansatz vor, der die Probleme aus dem Ansatz in Teil I umgeht. Wir haben eine Methode entwickelt, die es ermöglicht ganze Semantic Web Knowledge Graphs in einen Feature Space mit geringer Dimensionalität zu transformieren. Dabei wird jede Entität und Relation als numerischer Wert in einem Vektor repräsentiert. Nach der Projektion in einen solchen Vektorraum ist zu sehen, dass Entitäten, die sich semantisch ähnlich sind, auch nah beieinander im Vektorraum abgebildet werden. Hierbei haben wir mehrere Semantic Web Knowledge Graphs getestet, um zu zeigen, dass eine solche Darstellung der Entitäten vorteilhaft für verschiedene Data Mining Aufgaben ist. Außerdem können wir zeigen, dass solche Features auch auf anderen Datensätzen für andere Aufgaben genutzt werden können.

Im dritten Teil beschreiben wir eine Liste von Anwendungen, die Semantic Web Knowledge Graphs benutzen und dabei über Standardaufgaben des Data Mining wie Klassifizierung und Regression hinausgehen. Hierbei zeigen wir, dass die Ansätze aus Teil I und II in Anwendungen aus verschiedenen Domänen benutzt werden können. Speziell gehen wir dabei auf die Nutzung von Semantic Web Knowledge Graphs in der Analyse von Statistiken, zum Entwickeln von Empfehlungsdiensten, der Modellierung von Entitäten und Dokumenten und der Induktion von Taxonomien.

# Contents

# List of Publications

Parts of the work presented in this thesis have been previously published in international journals and proceedings of international conferences. For all publications the author of this thesis was the key contributor of the work presented in both the publications and this thesis.

**International Journals:**

- Petar Ristoski, Christian Bizer, Heiko Paulheim: *Mining the web of linked data with rapidminer.* Web Semantics: Science, Services and Agents on the World Wide Web, Vol 35, pages 142–151, 2015.
- Petar Ristoski, Heiko Paulheim: *Semantic Web in Data Mining and Knowledge Discovery: A Comprehensive Survey.* Web Semantics: Science, Services and Agents on the World Wide Web, Vol 36, pages 1–22, 2016.
- Petar Ristoski, Jessica Rosati, Tommaso Di Noia, Renato de Leone, Heiko Paulheim: *RDF2Vec: RDF Graph Embeddings and Their Applications.* The Semantic Web Journal – Accepted under minor revision.

**International Conferences:**

- Heiko Paulheim, Petar Ristoski, Evgeny Mitichkin, Christian Bizer: *Data mining with background knowledge from the web.* Proceedings of the 5th RapidMiner World, Boston, USA, pages 1–14, 2014.
- Petar Ristoski, Eneldo Loza Mencía, Heiko Paulheim: *A hybrid multi-strategy recommender system using linked open data.* Semantic Web Evaluation Challenge, Crete, Greece, May, 2014.
- Petar Ristoski, Heiko Paulheim: *Feature selection in hierarchical feature spaces.* Proceedings of the 17th International Conference on Discovery Science, Bled, Slovenia, October, 2014.
- Petar Ristoski, Heiko Paulheim: *Visual Analysis of Statistical Data on Maps Using Linked Open Data.* Revised Selected Papers of the 12th European Semantic Web Conference, Portoroz, Slovenia, May 2015

- Petar Ristoski: *Towards Linked Open Data Enabled Data Mining*. Proceedings of the 12th European Semantic Web Conference, Portoroz, Slovenia, May 2015
- Petar Ristoski, Michael Schuhmacher, Heiko Paulheim: *Using Graph Metrics for Linked Open Data Enabled Recommender Systems.* Proceedings of the 16th International Conference on Electronic Commerce and Web Technologies, Valencia, Spain, September, 2015.
- Petar Ristoski, Heiko Paulheim: *Rdf2Vec: RFG Graph Embeddings for Data Mining.* Proceedings of the 15th International Semantic Web Conference, Kobe, Japan, October, 2016.
- Petar Ristoski, Gerben Klaas Dirk de Vries, Heiko Paulheim: *A Collection of Benchmark Datasets for Systematic Evaluations of Machine Learning on the Semantic Web.* Proceedings of the 15th International Semantic Web Conference, Kobe, Japan, October, 2016.
- Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, Heiko Paulheim: *Biased Graph Walks for RDF Graph Embeddings.* Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics, Amantea, Italy, June, 2017.
- Petar Ristoski, Stefano Faralli, Simone Paolo Ponzetto, Heiko Paulheim: *Large-scale taxonomy induction using entity and word embeddings.* Proceedings of the International Conference on Web Intelligence, pages 81–87, 2017.

**International Workshops:**

- Petar Ristoski, Heiko Paulheim: *Analyzing statistics with background knowledge from linked open data.* Proceedings of the 1st International Workshop on Semantic Statistics co-located with 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, October 11th, 2013.
- Petar Ristoski, Heiko Paulheim: *A Comparison of Propositionalization Strategies for Creating Features from Linked Open Data.* Proceedings of the 1st Workshop on Linked Data for Knowledge Discovery co-located with European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2014), Nancy, France, September 19th, 2014.
- Petar Ristoski, Heiko Paulheim, Vojtech Svatek, and Vaclav Zeman: *Linked Data Mining Challenge 2015*. Proceedings of the 4th Workshop on Data Mining and Knowledge Discovery meets Linked Open Data, Portoroz, Slovenia, May 31, 2015..
- Petar Ristoski, Heiko Paulheim, Vojtech Svatek, and Vaclav Zeman: *Linked Data Mining Challenge 2016*. Proceedings of the 5th Workshop on Data Mining and Knowledge Discovery meets Linked Open Data, Heraklion, Greece, May 30th, 2016.

- Jessica Rosati, Petar Ristoski, Tommaso Di Noia, Renato de Leone, Heiko Paulheim: *RDF graph embeddings for content-based recommender systems.* Proceedings of the 3rd Workshop on New Trends in Content-based Recommender Systems, in conjunction with the 10th ACM Conference on Recommender Systems Boston, MA, USA, September 16, 2016.

**Papers published during the PhD studies that are not part of this thesis:**

- Benjamin Schäfer, Petar Ristoski, Heiko Paulheim: *What is special about Bethlehem, Pennsylvania?: identifying unexpected facts about DBpedia entities* CEUR workshop proceedings, 2015.
- Axel Schulz, Frederik Janssen, Petar Ristoski, Johannes Fürnkranz: *Event-Based Clustering for Reducing Labeling Costs of Event-related Microposts.* Proceedings of the 9th International AAAI Conference on Web and Social Media, May, 2015.
- Axel Schulz, Petar Ristoski, Johannes Fürnkranz, Frederik Janssen: *Event-based Clustering for Reducing Labeling Costs of Incident-Related Microposts* Proceedings of the 2nd International Workshop on Mining Urban Data co-located with 32nd International Conference on Machine Learning, 2015.
- Oliver Lehmberg, Dominique Ritze, Petar Ristoski, Robert Meusel, Heiko Paulheim, Christian Bizer: *The mannheim search join engine.* Web Semantics: Science, Services and Agents on the World Wide Web, Vol 35, pages 159–1166, 2015.
- Petar Ristoski, Peter Mika: *Enriching Product Ads with Metadata from HTML Annotations.* Proceedings of the 13th International Conference on The Semantic Web, Heraklion, Greece, May, 2016.
- Anna Lisa Gentile, Petar Ristoski, Steffen Eckel, Dominique Ritze, Heiko Paulheim: *Entity Matching on Web Tables: a Table Embeddings approach for Blocking.* Proceedings of the 20th International Conference on Extending Database, March, 2017.
- Mehwish Alam, Diego Reforgiato Recupero, Misael Mongiovi, Aldo Gangemi, Petar Ristoski: *Event-Based Knowledge Reconciliation using Frame Embeddings and Frame Similarity.* Journal of Knowledge-Based Systems, September, 2017.
- Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, Heiko Paulheim:: *Global RDF Vector Space Embeddings.* Proceedings of the 16th International Semantic Web Conference, Vienna, Austria, October, 2017.
- Petar Ristoski, Petar Petrovski, Petar Mika, Heiko Paulheim: *A Machine Learning Approach for Product Matching and Categorization.* The Semantic Web Journal

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Data mining is defined as "a non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data" [85], or "the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner" [108]. As such, data mining and knowledge discovery (KDD) are typically considered knowledge intensive tasks. Thus, knowledge plays a crucial role here. Knowledge can be (a) in the primary data itself, from where it is discovered using appropriate algorithms and tools, (b) in external data, which has to be included with the problem first (such as background statistics or master file data not yet linked to the primary data), or (c) in the data analyst's mind only.

The latter two cases are interesting opportunities to enhance the value of the knowledge discovery processes. Consider the following case: a dataset consists of countries in Europe and some economic and social indicators. There are, for sure, some interesting patterns that can be discovered in the data. However, an analyst dealing with such data on a regular basis will know that some of the countries are part of the European Union, while others are not. Thus, she may add an additional variable `EU_Member` to the dataset, which may lead to new insights (e.g., certain patterns holding for EU member states only).

In that example, knowledge has been added to the data from the analyst's mind, but it might equally well have been contained in some exterior source of knowledge, such as Semantic Web knowledge graphs. Semantic Web knowledge graphs are a backbone of many information systems that require access to structured knowledge. Those knowledge graphs contain factual knowledge about real world entities and their relations and attributes in a fully machine-readable format. Following the principles of the Semantic Web [13], such knowledge graphs are publicly available as Linked Open Data [21]. Linked Open Data (LOD) is an open, interlinked collection of datasets in machine-interpretable form, covering multiple domains from life sciences to government data [267]. Some of the most used Semantic Web knowledge bases are DBpedia [9], YAGO [291], and Wikidata [319]. In the last decade, a vast amount of approaches have been proposed that combine

methods from data mining and knowledge discovery with Semantic Web knowledge graphs. The goal of those approaches is to support different data mining tasks, or to improve the Semantic Web itself.

In their seminal paper from 1996, Fayyad et al. [85] introduced a process model for knowledge discovery processes. The model comprises five steps, which lead from raw data to actionable knowledge and insights which are of immediate value to the user, which comprises five steps: *Selection*, *Preprocessing*, *Transformation*, *Data Mining*, and *Evaluation and Interpretation*. It has been shown that Semantic Web knowledge graphs can support each step of the KDD pipeline. Given a set of local data (such as a relational database), the first step is to link the data to the corresponding knowledge graph concepts from the chosen LOD dataset. Once the local data is linked to a LOD dataset, we can explore the existing links in the dataset pointing to the related entities in other LOD datasets. In the next step, various techniques for data consolidation, preprocessing and cleaning are applied, e.g., schema matching, data fusion, value normalization, treatment of missing values and outliers. Next, some transformations on the collected data need to be performed in order to represent the data in a way that it can be processed with any arbitrary data analysis algorithms. Since most algorithms demand a propositional form of the input data, this usually includes a transformation of the graph-based LOD data to a canonical propositional form. After the data transformation is done, a suitable data mining algorithm is selected and applied on the data. In the final step, the results of the data mining process are presented to the user. Here, to ease the interpretation and evaluation of the results of the data mining process, Semantic Web and LOD can be used as well.

To clarify the process of exploiting Semantic Knowledge graphs, we introduce a use case of exploiting Semantic Web knowledge graphs in recommneder systems. Recommender systems have changed the way people find and buy products and services. As the Web has grown over time, and the number of products and services within, recommender systems represent a powerful method for users to filter that large information and product space. With the introduction of the Semantic Web and Linked Open Data, recommender systems are emerging research area that extensively use Semantic Web knowledge graphs as background knowledge for extracting useful data mining features that could improve the recommendation results. It has been shown that Linked Open Data can improve recommender systems towards a better understanding and representation of user preferences, item features, and contextual signs they deal with. Knowledge graphs have been used in content-based, collaborative, and hybrid techniques, in various recommendation task, i.e., rating prediction, Top-N recommendations, cross-domain recommendation and diversity in content-based recommendations.

Consider we are given a dataset of books with a set of user ratings. The task is to predict the rating for each book by each user. We have to note that some books might not be rated by any users, and some users might not have rated any books. This is also known as the *cold start problem*, for which standard collaborative filtering recommender systems cannot predict the ratings for such books

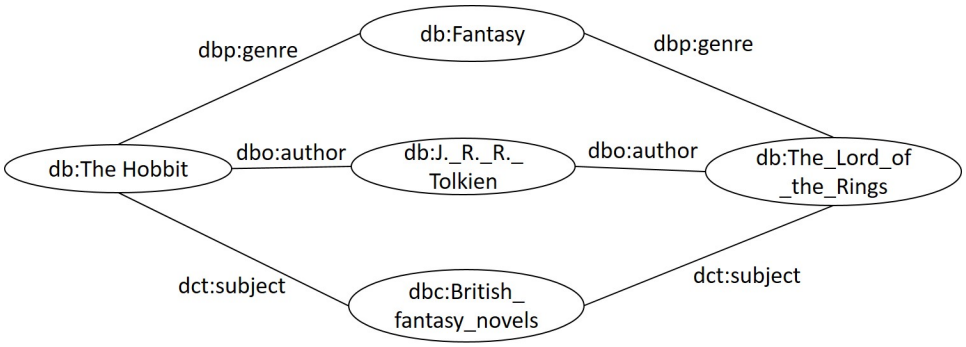



Figure 1.1: An excerpt of the DBpedia graph for the books "The Lord of the Rings" and "The Hobit"

and users, because there is no sufficient information in the dataset. In such cases, content-based recommender systems have to be used. However, to be able to build content-based recommenders, we have to extract content features for each of the books. Semantic Web knolwedge graphs, like DBpedia, contain structured information about books that can be automatically extracted and utilized. For example, in DBpedia we can easily extract the author of the book (*dbo:author*), the language of the book (*dbo:language*), the genre of the book (*dbp:genre*), and many more properties that can be used to build a recommender system. By analyzing the already rated books by a user, the recommender can identify some patterns in those books and make useful recommendations for new books for that user, e.g., if a user has already rated positively the book "The Lord of the Rings", a recommender system might recommend the book "The Hobit". The system can then easily give an explanation to why the book was recommended to the user by displaying the most important shared relations for these two books, e.g., both books are "High fantasy", both books are written by the same author "J. R. R. Tolkien", and both books belong to the same category "British fantasy novels". An excerpt of the DBpedia graph for these two books is given in Figure 1.1.

To be able to make use of the structured knowledge in DBpedia for the given task, we have to first link the dataset of books to the corresponding DBpedia entities. To do so, we use the name of the books (and if there are any further information, e.g., the author of the book), as given in the dataset, to match the books to the corresponding entities in DBpedia. Once we have set the links to DBpedia, we can automatically link the local dataset to other knowledge graphs simply by following the *owl:sameAs* links in DBpedia, e.g., YAGO, Wikidata, or different language versions of DBpedia.

In the next step, we extract useful features that would be used to describe the books. This is the *Transformation* step, which has been shown to be the most important step of the KDD pipeline, i.e., being able to select the most informative and representative features for a given dataset is the key step for building high performance data mining models [75]. As most of the standard data min-

ing algorithms and tools require propositional feature vector representation, for making use of Semantic Web knowledge graphs in the data mining step transformations have to be performed, which create propositional features from the graphs, i.e., a process called *propositionalization* [154]. One approach would be to ask a domain expert to manually select the features that can be used for the given task, e.g., as we already mentioned, the author, the genre, the language of the book might be good features for describing the book. However, this approach is time consuming and costly. Another approach is to automatically retrieve features from the knowledge base, e.g., as we propose in Part I of this thesis, we can use unsupervised propositinalization strategies from types and relations in knowledge graphs. Such strategies automatically transform the graph into feature vectors, by flattening down the incoming and outgoing relations for each book. For example, we can automatically extract all the types and categories for the book "The Lord of the Rings": "dbc:Adventure_novels", "dbc:British_novels-_adapted_into_films", "dbc:British_adventure_novels", etc. In Part I, we give an overview of a set of propositionalization strategies that exploit the relations and the structure of the graph to generate propositional feature vectors, in a complete unsupervised manner, independently of the task and the dataset. However, such feature strategies do not scale when the input dataset is large, i.e., the number of generated features for all the books can quickly become unmanageable. Therefore, there is a need for feature selection approaches. In Part I, we describe a feature selection approach that exploits the graph structure in order to select only the most informative and most representative features, while significantly reducing the number of features in the complete dataset. For example, in the previous example, the category "dbc:Adventure_novels" subsumes the category "dbc:British_adventure_novels", therefore we can remove the latter one.

While this feature engineering approach follows the standard KDD pipeline, there might be a more efficient approach for transforming the graph into propositional feature vectors. In Part II, we introduce more sophisticated propositinalization strategies that embed the whole structure of the knowledge graph into a low dimensional feature space, i.e., knowledge graph embeddings, where each entity and relation in the knowledge graph is represented as a numerical vector. Such approaches leverage local information from graph sub-structures, harvested by graph kernels and graph walks, to learn latent numerical representations of entities in RDF graphs, i.e., the approach explicitly models the assumption that entities that appear in similar graph sub-structures in the knowledge graph, are semantically similar to each other. Projecting such latent representations of entities into a lower dimensional feature space shows that semantically similar entities appear closer to each other. For example, the books "The Lord of the Rings" and "The Hobit", would have very similar feature representation. Using such graph embedding techniques, circumvents the need for manual feature engineering and applying feature selection approaches. The generated feature vectors have the same size, independently of the given task or dataset, i.e., the embeddings are generated once for the whole knowledge graph and can be reused in different applications, as shown in

Part III of this thesis.

In the final step of our recommender system use case, we build the actual recommender system using the previously generated features. To do so, standard recommender methods can be used, for building content-based, collaborative or hybrid recommender systems.

In this thesis, we focus on the third step of knowledge discovery process, *Transformation*, proposing multiple feature propositionalization strategies, which are evaluated on a large set of datasets, and applied in real world applications.

## 1.1 Research Questions

The goal of this thesis is to answer *How can Semantic Web knowledge graphs be exploited as background knowledge in data mining?*. To answer this question, we articulated our research in a specific set of sub-questions, which are summarized as:

- **RQ1**: *Do existing Semantic Web knowledge graphs cover enough topics to be used in different data mining tasks?* Data mining tasks cover various topic domains, e.g., entertainment, medicine, geography, science, etc. Therefore, to use Semantic Web knowledge graphs in data mining, the graphs need to contain data about various topic domains. This question is addressed in Chapter 4.

- **RQ2**: *How can we automatically discover and integrate useful background information from Semantic Web knowledge graphs for data mining?* After we have shown that Semantic Web knowledge graphs contain valuable data for various data mining tasks, we need to develop approaches for accessing this data. First, we have to develop approaches for automatic discovery of useful data from Semantic Web knowledge graphs for a given data mining task. Then, we have to perform transformation of the graph data in order to be able to use it with the standard data mining tools and algorithms. And lastly, we have to perform feature selection, in order to select the most informative and representative features that will lead to better performance of the data mining algorithms. This question is addressed throughout Part I of this thesis.

- **RQ3**: *How can we efficiently and effectively reuse the knowledge from the Semantic Web knowledge graphs independently of the data and the data mining task?* The existing Semantic Web knowledge graphs are rather big, therefore approaches for efficient discovery and extraction of background knowledge are needed. Such approaches, should be able to convert the graph into data mining features that can be readily reused independently of the data mining task and dataset. This question is addressed throughout Part II of this thesis.

- **RQ4**: *What are the real-world data mining applications that benefit from adding background knowledge from Semantic Web knowledge graphs?* Once the approaches and the tools for accessing background knowledge from Semantic Web

knowledge graphs have been developed, we have to identify what are the different real-world data mining applications that can benefit from such knowledge. Data mining is a broad area, covering many tasks and applications, starting from the basic classification and regression tasks to recommender systems, which are becoming more and more important in the research and business area. This question is addressed throughout Part III of this thesis.

## 1.2 Contributions

The contribution of this thesis is broad and diverse, showing the potential of Semantic Web knowledge graphs as background knowledge in data mining. In particular, this thesis makes the following contributions:

- In-depth overview and comparison of existing approaches for exploiting Semantic Web knowledge graph in each step of the knowledge discovery pipeline, and data mining in general - Chapter 3
- A collection of benchmark datasets for systematic evaluations of machine learning on the Semantic Web - Chapter 4
- Empirical evaluation of propositionalization strategies for generating features from Semantic Web knowledge graphs - Chapter 5
- An approach for feature selection in hierarchical feature spaces - Chapter 6
- A tool for mining the web of Linked Data, which provides approaches for generating, selecting and integrating machine learning features from many LOD sources - Chapter 7
- An approach for Semantic Web knowledge graphs embeddings, and their applications in data mining - Chapter 8 and Chapter 9
- A list of developed applications that exploit Semantic Web knoweldge grpahs, i.e., a tool for analyzing statistics with background knowledge from Semantic Web knowledge graphs - Chapter 10; Three approaches for exploiting Semantic Web knowledge graphs for building recommender systems - Chapter 11; An approach for entities and document modeling - Chapter 12; and an approach for taxonomy induction - Chapter 13

## 1.3 Structure

In this section, the content of each following chapter is summarized:

**Chapter 2: Fundamentals** After introducing and motivating the goal of the thesis, this chapter, presents an overview of the basics of the knowledge discovery process, Semantic Web, Linked Open Data, Semantic Web knowledge graphs, and commonly used Semantic Web knowledge graphs.

**Chapter 3: Related Work** This chapter gives a comprehensive overview of existing data mining approaches exploiting Semantic Web knowledge graphs in different stages of the knowledge discovery process.

**Chapter 4: A Collection of Benchmark Datasets for Systematic Evaluations of Machine Learning on the Semantic Web** This chapter introduced a collection of datasets for benchmarking machine learning approaches for the Semantic Web. Such a collection of datasets can be used to conduct quantitative performance testing and systematic comparisons of approaches. The collection of dataset is also used for evaluating the approaches described in the following chapters.

**Chapter 5: Propositionalization Strategies for Creating Features from Linked Open Data** This chapter describes a set of strategies for creating features from types and relations in Semantic Web knowledge graphs. Furthermore, it conducts an empirical evaluation of the propositionalization strategies on a number of different datasets and across different tasks, i.e., classification, regression, and outlier detection.

**Chapter 6: Feature Selection in Hierarchical Feature Spaces** The experiments conducted in the previous chapter showed that the number of generated features from knowledge graphs can rapidly increase as the size of the input dataset increases. Given that knowledge graphs are usually backed by hierarchical structures, this chapter describes an approach that exploits hierarchies for feature selection in combination with standard metrics.

**Chapter 7: Mining the Web of Linked Data with RapidMiner** This chapter discusses how the Web of Linked Data can be mined using the full functionality of *RapidMiner*. The chapter introduces an extension to RapidMiner, which allows for bridging the gap between the Web of Data and data mining, and which can be used for carrying out sophisticated analysis tasks on and with Linked Open Data.

**Chapter 8: RDF2Vec: RDF Graph Embeddings for Data Mining** The previous chapters of the thesis introduced several strategies for feature generation from Semantic Web knowledge graphs. However, such feature strategies do not scale when the input dataset is large, i.e., the number of generated features quickly becomes unmanageable and there is a need for feature selection. This chapter describes an approach for Semantic Web knowledge graphs embedding. The chapter

shows that such embeddings can be used as feature vectors for machine learning tasks, like classification and regression.

**Chapter 9: Biased Graph Walks for RDF Graph Embeddings** This chapter extends the previous chapter, by examining methods to direct the random walks in more meaningful ways, i.e., being able to capture more important information about each entity in the graph. The chapter evaluates a dozen weighting schemes which influence the walks and, thus, the resulting sequences.

**Chapter 10: Analyzing Statistics with Background Knowledge from Semantic Web Knowledge Graphs** Statistical datasets are widely spread and published on the Web, however the availability of tools for analyzing such data is limited. This chapter presents the Web-based tool *ViCoMap*, which allows automatic correlation analysis and visualizing statistical data on maps using Semantic Web knowledge graphs.

**Chapter 11: Semantic Web enabled Recommender Systems** Since the Linked Open Data has a coverage of a large number of domains, it is a suitable source for building content-based recommender systems. This chapter presents three approaches for exploiting Semantic Web knowledge graphs for building recommender systems. The first approach is based on graph metrics, the second approach is based on a hybrid approach using flat features extracted from Semantic Web knowledge graphs, and the third approach uses graph embeddings as features to build a content-based recommender system.

**Chapter 12: Entity and Document Modeling using Semantic Web Graph Embeddings** This chapter shows that the graph embeddings, introduced in Chapter 8, can be used for the task of entity and document modeling, which are fundamental problems in numerous tasks in information retrieval, natural language processing, and Web-based knowledge extraction.

**Chapter 13: Taxonomy Induction Using Knowledge Graph Embeddings** As shown in the previous chapters, it is crucial to define a high quality class hierarchy for a knowledge base in order to allow effective access to the knowledge base from various Natural Language Processing, Information Retrieval, and any Artificial Intelligence systems and tools. Thus, this chapter presents an approach that makes use of the graph embeddings introduced in Chapter 8 for automatic unsupervised class subsumption axiom extraction from semi-structured knowledge bases.

# Chapter 2

# Fundamentals

This chapter introduces the basics of the Semantic Web, Linked Open Data, Semantic Web knowledge graphs, the knowledge discovery process, and Semantic Web enabled knowledge discovery process.

## 2.1 Semantic Web Knowledge Graphs

The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. Semantic Web technologies facilitate building a large-scale Web of machine-readable and machine-understandable knowledge, and thus facilitate data reuse and integration. Since the beginning, the Semantic Web has promoted a graph-based representation of knowledge, e.g., using the Resource Description Framework (RDF)[1]. In general, RDF is a framework which provides capabilities to describe information about resources. As defined by the W3C recommendation for RDF[2], the core structure of RDF is a set of triples, each consisting of a subject, a predicate and an object, e.g. *db:Berlin dbo:capitalOf db:Germany* represents a triple. A set of such triples is called an RDF graph. In such a graph-based knowledge representation, *entities*, which are the nodes of the graph, are connected by *relations*, which are the edges of the graph. Each entity is uniquely identified with a *Internationalized Resource Identifier (IRI)*, and usually entities have types, denoted by *is a* relations, e.g. *db:Berlin rdf:type dbo:City*.

We give a formal definition of an RDF graph in Definition 1.

**Definition 1.** *An RDF graph is a labeled graph G = (V, E), where V is a set of vertices, and E is a set of directed edges, where each vertex $v \in V$ is identified by a unique identifier, and each edge $e \in E$ is labeled with a label from a finite set of edge labels.*

[1] `https://www.w3.org/RDF/`

[2] `https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/`

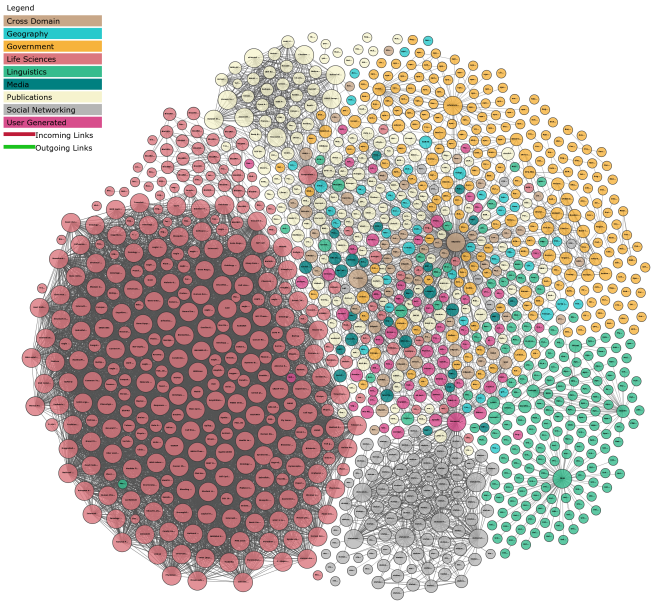



Figure 2.1: The Linking Open Data cloud diagram 2017

Paulheim [223] provides a more loose definition of a Semantic Web knowledge graph, where a knowledge graph:

- mainly describes real world entities and their interrelations, organized in a graph.
- defines possible classes and relations of entities in a schema.
- allows for potentially interrelating arbitrary entities with each other.
- covers various topical domains.

### 2.1.1 Linked Open Data

The basics of the Semantic Web were set by Sir Tim Berners-Lee in 2001 [13], which later lead to the creation of the Linked Opend Data [21] [3]. Linked Open Data (LOD) is an open, interlinked collection of datasets in machine-interpretable form, covering multiple domains from life sciences to government data [267]. Currently, around 1,000 datasets are interlinked in the Linked Open Data cloud, with the majority of links connecting identical entities in two datasets. The structure of the Linked Open Data cloud is given in Figure 2.1.[4]

The three most commonly used large cross-domain knowledge graphs in the LOD cloud are: DBpedia [9], YAGO [291], and Wikidata [319]. Also, these three knowledge graphs are heavily used in this thesis.

[3] `https://www.w3.org/DesignIssues/LinkedData.html`
[4] Available at: `http://lod-cloud.net/`

**DBpedia:** DBpedia is the central hub of the LOD cloud. DBpedia is a knowledge graph which is extracted from structured data in Wikipedia, and makes this information available on the Web in various machine-readable formats.[5] The main source for this extraction are Wikipedia infoboxes, which are the property summarizing tables found on most of the Wikipedia pages. In a comunity based project, types of the Wikipedia infoboxes are mapped to the DBpedia ontology, and the keys used in those infoboxes are mapped to properties in the DBpedia ontology. Based on those mappings, the DBpedia knowledge graph is extracted [162].

**YAGO (Yet Another Great Ontology):** Like DBpedia, YAGO is also extracted from Wikipedia. The YAGO ontology is built from the category system in Wikipedia and the lexical resource WordNet [180], with infobox properties manually mapped to a fixed set of attributes.

**Wikidata:** Wikidata is a collaboratively edited knowledge graph, operated by the Wikimedia foundation[6] that many language editions of Wikipedia.

## 2.2 Data Mining and The Knowledge Discovery Process

Data mining is defined as "a non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data" [85], or "the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner" [108]. On a more general level, the data mining field is concerned with the development of methods and algorithms for discovering patterns in large data sets for better understanding of the data.

In a seminal paper from 1996, Fayyad et al. [85] introduced a process model for knowledge discovery processes. The model consists of five steps, which lead from raw data to actionable knowledge and insights which are of immediate value to the user. The whole process is shown in Figure 2.2. It comprises five steps:

1. *Selection* The first step is developing an understanding of the application domain, capturing relevant prior knowledge, and identifying the data mining goal from the end user's perspective. Based on that understanding, the target data used in the knowledge discovery process can be chosen, i.e., selecting proper data samples and a relevant subset of variables.

2. *Preprocessing* In this step, the selected data is processed in a way that allows for a subsequent analysis. Typical actions taken in this step include the handling of missing values, the identification (and potentially correction) of noise and errors in the data, the elimination of duplicates, as well

[5] `http://www.dbpedia.org`
[6] `http://wikimediafoundation.org/`

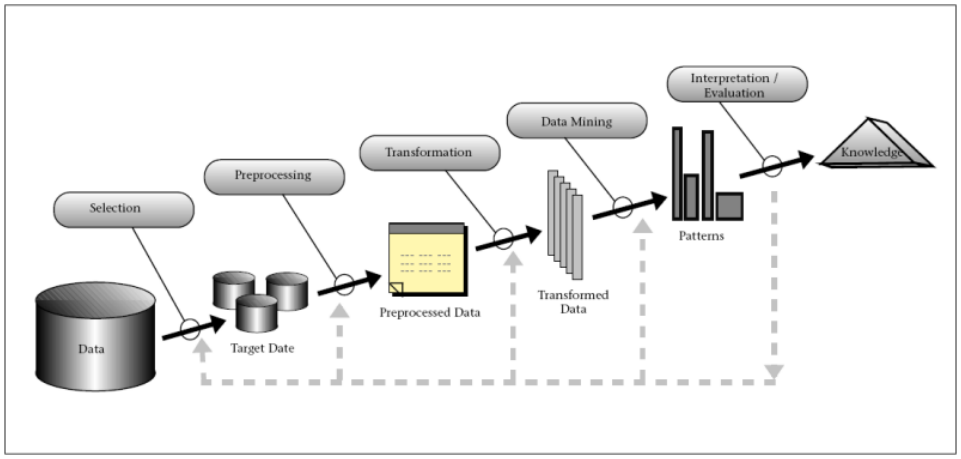



Figure 2.2: An Overview of the Steps That Compose the KDD Process

as the matching, fusion, and conflict resolution for data taken from different sources.

3. *Transformation* The third step produces a projection of the data to a form that data mining algorithms can work on – in most cases, this means turning the data into a propositional form, where each instance is represented by a feature vector. To improve the performance of subsequent data mining algorithms, dimensionality reduction methods can also be applied in this step to reduce the effective number of variables under consideration.

4. *Data Mining* Once the data is present in a useful format, the initial goal of the process is matched to a particular method, such as classification, regression, or clustering. This step includes deciding which models and parameters might be appropriate (for example, models for categorical data are different than models for numerical data), and matching a particular data mining method with the overall criteria of the KDD process (for example, the end user might be more interested in an interpretable, but less accurate model than a very accurate, but hard to interpret model). Once the data mining method and algorithm are selected, the data mining takes place: searching for patterns of interest in a particular representational form or a set of such representations, such as rule sets or trees.

5. *Evaluation and Interpretation* In the last step, the patterns and models derived by the data mining algorithm(s) are examined with respect to their validity. Furthermore, the user assesses the usefulness of the found knowledge for the given application. This step can also involve visualization of the extracted patterns and models, or visualization of the data using the extracted models.

The quality of the found patterns depends on the methods being employed in

each of these steps, as well as their interdependencies. Thus, the process model foresees the possibility to go back to each previous step and revise decisions taken at that step, as depicted in Figure 2.2. This means that the overall process is usually repeated after adjusting the parametrization or even exchanging the methods in any of these steps until the quality of the results is sufficient.

## 2.3 Semantic Web Knowledge Graphs in Data Mining and Knowledge Discovery

In the last decade, a vast amount of approaches have been proposed which combine methods from data mining and knowledge discovery with Semantic Web knowledge graphs. The goal of those approaches is to support different data mining tasks, or to improve the Semantic Web itself. All those approaches can be divided into three broader categories:

- Using Semantic Web based approaches, Semantic Web Technologies, and Linked Open Data to support the process of knowledge discovery.
- Using data mining techniques to mine the Semantic Web, also called *Semantic Web Mining*.
- Using machine learning techniques to create and improve Semantic Web data.

Stumme et al. [289] have provided an initial survey of all three categories, later focusing more on the second category. Dating back to 2006, this survey does not reflect recent research works and trends, such as the advent and growth of Linked Open Data. More recent surveys on the second category, i.e., Semantic Web Mining, have been published by Sridevi et al [285], Quoboa et al. [242], Sivakumar et al. [280], and Dou et al. [73].

Tresp et al. [306] give an overview of the challenges and opportunities for the third category, i.e., machine learning on the Semantic Web, and using machine learning approaches to support the Semantic Web. The work has been extended in [244].

In this thesis, we focus on the first category, i.e., the usage of Semantic Web knowledge graphs and Linked Open Data to support and improve data mining and knowledge discovery

Many approaches have been proposed in the recent past for using Semantic Web knowledge graphs in data mining processes, for various purposes, such as the creation of additional variables. Following the well-known data mining process model proposed by Fayyad et al. [85], we discuss how semantic data is exploited at the different stages of the data mining model. Furthermore, we analyze how different characteristics of Linked Open Data, such as the presence of interlinks between datasets and the usage of ontologies as schemas for the data, are exploited by the different approaches.

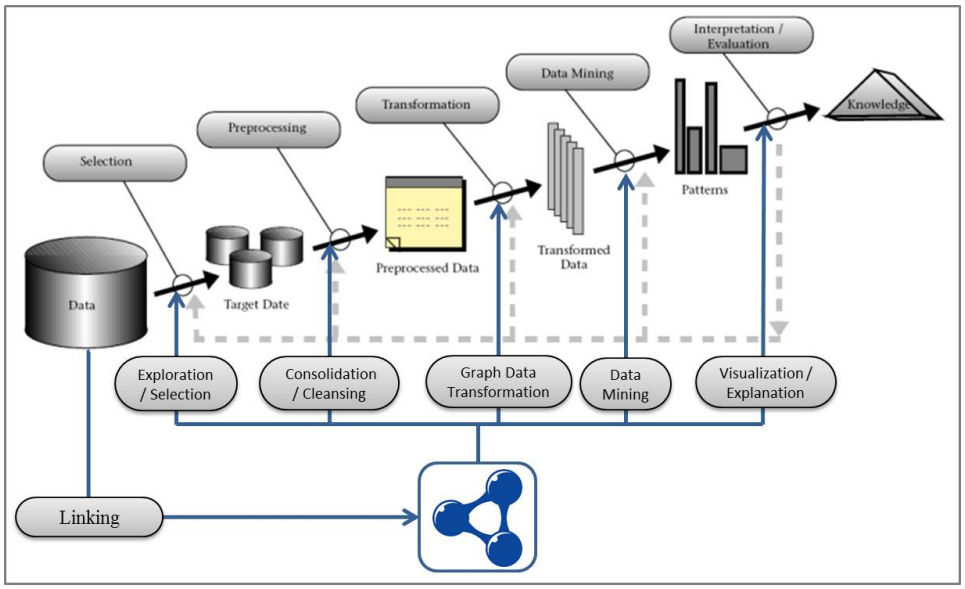



Figure 2.3: An Overview of the Steps of the Linked Open Data enabled KDD pipeline.

As a means to express knowledge about a domain in the Semantic Web, *ontologies* have been introduced in the early 1990s as "explicit formal specifications of the concepts and relations among them that can exist in a given domain" [102]. For the area of knowledge discovery and data mining, Nigoro et al. [199] divide ontologies used in this area into three categories:

- *Domain ontologies:* Express background knowledge about the application domain, i.e., the domain of the data at hand on which KDD and data mining is performed.
- *Ontologies for data mining process:* Define knowledge about the data mining process, its steps and algorithms and their possible parameters.
- *Metadata ontologies:* Describe meta knowledge about the data at hand, such as provenance information, e.g., the processes used to construct certain datasets.

It has been already shown that ontologies for the data mining process and metadata ontologies can be used in each step of the KDD process. However, we want to put a stronger focus on the usage of Linked Open Data (LOD) in the process of knowledge discovery, which represents a publicly available interlinked collection of datasets from various topical domains [21, 268].

Figure 2.3 gives an overview of the Linked Open Data enabled knowledge discovery pipeline. Given a set of local data (such as a relational database), the first step is to link the data to the corresponding LOD concepts from the chosen

LOD dataset (cf. section 3.1)[7]. Once the local data is linked to a LOD dataset, we can explore the existing links in the dataset pointing to the related entities in other LOD datasets. In the next step, various techniques for data consolidation, preprocessing and cleaning are applied, e.g., schema matching, data fusion, value normalization, treatment of missing values and outliers, etc. (cf. section 3.2). Next, some transformations on the collected data need to be performed in order to represent the data in a way that it can be processed with any arbitrary data analysis algorithms (cf. section 3.3). Since most algorithms demand a propositional form of the input data, this usually includes a transformation of the graph-based LOD data to a canonical propositional form. After the data transformation is done, a suitable data mining algorithm is selected and applied on the data (cf. section 3.4). In the final step, the results of the data mining process are presented to the user. Here, ease the interpretation and evaluation of the results of the data mining process, Semantic Web and LOD can be used as well (cf. section 3.5).

[7]We should note that the data can be linked to the LOD datasets in different stages of the KDD process, for example, in some approaches only the results and the discovered patterns from the data mining process are linked to a given LOD dataset in order to ease the interpretation of them. For simplicity's sake we describe the process of linking as the first step, which is also depicted in Figure 2.3.

# Chapter 3

# Related Work

In this chapter, we give a comprehensive overview of existing data mining approaches exploiting Semantic Web knowledge graphs in different stages of the knowledge discovery process, as discussed in Chapter 2.

For the overview of LOD enabled approaches, we have selected only approaches that fulfill the following criteria:

1. They are designed and suitable for improving the KDD process in at least one step
2. They make use of one or more datasets on the Semantic Web

Each of the approaches is assessed using a number of criteria:

1. Is the approach domain-independent or tailored to a specific domain?
2. Is the approach tailored towards a specific data mining technique (e.g., rule induction)?
3. Does it use a complex ontology or only a weakly axiomatized one (such as a hierarchy)?
4. Is any reasoning involved?
5. Are links to other datasets (a core ingredient of Linked Open Data) used?
6. Are the semantics of the data (i.e., the ontology) exploited?

Furthermore, we analyze which Semantic Web datasets are used in the papers, to get a grasp of which are the most prominently used ones.

In the following sections, we introduce and discusse the individual approaches[1]. A small box at the end of each section gives a brief summary, a coarse-grained comparison, and some guidelines for data mining practitioners who want to use the approaches in actual projects.

[1]We should note that some of the approaches might be applicable in several steps of the LOD-enabled KDD pipeline. However, in almost all cases, there is one step which is particularly in the focus of that work, and we categorize those works under that step.

**The work presented in this chapter has been published before as: "Petar Ristoski, Heiko Paulheim: *Semantic Web in Data Mining and Knowledge Discovery: A Comprehensive Survey.* Web Semantics: Science, Services and Agents on the World Wide Web, Vol 36, pages 1–22, 2016." [255].**

## 3.1 Selection

To develop a good understanding of the application domain, and the data mining methods that are appropriate for the given data, a deeper understanding of the data is needed. First, the user needs to understand what is the domain of the data, what knowledge is captured in the data, and what is the possible additional knowledge that could be extracted from the data. Then, the user can identify the data mining goal more easily, and select a sample of the data that would be appropriate for reaching that goal.

However, the step of understanding the data is often not trivial. In many cases, the user needs to have domain specific knowledge in order to successfully understand the data. Furthermore, the data at hand is often represented in a rather complex structure that contains hidden relations.

To overcome this problem, several approaches propose using Semantic Web techniques for better representation and exploration of the data, by exploiting domain specific ontologies and Linked Open Data. This is the first step of the Semantic Web enhanced KDD pipeline, called *linking*. In this step, a *linkage*, or *mapping*, to existing ontologies, and LOD datasets is performed on the local data.

Once the linking is done, additional background knowledge for the local data can be automatically extracted. That allows to formally structure the domain concepts and information about the data, by setting formal types, and relations between concepts. Using background knowledge in many cases the users can easily understand the data domain, without the need for employing domain experts.

Furthermore, many tools for visualization and exploration of LOD data exist that would allow an easier and deeper understanding of the data. An overview of tools and approaches for visualization and exploration of LOD is given in the survey by Dadzie et al. [49]. The authors first set the requirements or what is expected of the tools for visualization or browsing the LOD: (i) the ability to generate an overview of the underlying data, (ii) support for filtering out less important data in order to focus on selected regions of interest (ROI), and (iii) support for visualizing the detail in ROIs. Furthermore, all this tools should allow the user to intuitively navigate through the data, explore entities and relations between them, explore anomalies within the data, perform advanced querying, and data extraction for reuse. They divided the analyzed browsers between those offering a text-based presentation, like Disco[2] and Sig.ma [308] and Piggy Bank [131], and those with

[2] `http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco`

visualization options, like Fenfire [112], IsaViz [3], and RelFinder[4]. The analysis of the approaches shows that most of the text-based browsers provide functionalists to support the tech-users, while the visualization-based browser are mostly focused on the non-tech users. Even though the authors conclude that there is only a limited number of SW browsers available, we can still make use of them to understand the data better and select the data that fits the data analyst's needs. The categorization of approaches in the survey by Dadzie et al. has been extended by by Peña et al. [230], based on the data datatypes that are visualized and the functionality needed by the analysts. The authors list some more recent approaches for advanced LOD visualization and exploration, like CODE [191], LDVizWiz [8], LODVisualization [32], and Payola [150].

The approaches for linking local data to LOD can be divided into three broader categories, based on the initial structural representation of the local data:

### 3.1.1 Using LOD to interpret relational databases

Relational databases are considered as one of the most popular storage solutions for various kinds of data, and are widely used. The data represented in relational databases is usually backed by a schema, which formally defines the entities and relations between them. In most of the cases, the schema is specific for each database, which does not allow for automatic data integration from multiple databases. For easier and automatic data integration and extension, a global shared schema definition should be used across databases.

To overcome this problem, many approaches for mapping relational databases to global ontologies and LOD datasets have been proposed. In recent surveys [310, 263, 284] the approaches have been categorized in several broader categories, based on three criteria: existence of an ontology, domain of the generated ontology, and application of database reverse engineering. Additionally, [284] provides a list of the existing tools and frameworks for mapping relational databases to LOD, from which the most popular and most used is the D2RQ tool [19]. D2RQ is a declarative language to describe mappings between application-specific relational database schemata and RDF-S/OWL ontologies. Using D2RQ, Semantic Web applications can query a non-RDF database using RDQL, publish the content of a non-RDF database on the Semantic Web using the RDF Net API[5], do RDFS and OWL inferencing over the content of a non-RDF database using the Jena ontology API[6], and access information in a non-RDF database using the Jena model API[7]. D2RQ is implemented as a Jena graph, the basic information representation object within the Jena framework. A D2RQ graph wraps one or more local relational

[3] http://www.w3.org/2001/11/IsaViz/

[4] http://www.visualdataweb.org/relfinder.php

[5] http://wifo5-03.informatik.uni-mannheim.de/bizer/rdfapi/tutorial/netapi.html

[6] https://jena.apache.org/documentation/ontology/

[7] https://jena.apache.org/tutorials/rdf_api.html

databases into a virtual, read-only RDF graph. D2RQ rewrites RDQL queries and Jena API calls into application-datamodel-specific SQL queries. The result sets of these SQL queries are transformed into RDF triples which are passed up to the higher layers of the Jena framework.

### 3.1.2 Using LOD to interpret semi-structured data

In many cases, the data at hand is represented in a semi structured representation, meaning that the data can be easily understand by humans, but it cannot be automatically processed by machines, because it is not backed by a schema or any other formal representation. One of the most used semi-structure representations of data is the tabular representation, found in documents, spreadsheets, on the Web or databases. Such representation often follows a simple structure, and unlike relational databases, there is no explicit representation of a schema.

Evidence for the semantics of semi-structured data can be found, e.g., in its column headers, cell values, implicit relations between columns, as well as caption and surrounding text. However, general and domain-specific background knowledge is needed to interpret the meaning of the table.

Many approaches have been proposed for extracting the schema of the tables, and mapping it to existing ontologies and LOD. Mulwad et al. have made significant contribution for interpreting tabular data using LOD, coming from independent domains [189, 188, 294, 185, 186, 187]. They have proposed several approaches that use background knowledge from the Linked Open Data cloud, like Wikitology [87], DBpedia [9], YAGO [291], Freebase [26] and WordNet [180], to infer the semantics of column headers, table cell values and relations between columns and represent the inferred meaning as graph of RDF triples. A table's meaning is thus captured by mapping columns to classes in an appropriate ontology, linking cell values to literal constants, implied measurements, or entities in the LOD cloud and identifying relations between columns. Their methods range from simple index lookup from a LOD source, to techniques grounded in graphical models and probabilistic reasoning to infer meaning associated with a table [187], which are applicable on different types of tables. i.e, relational tables, quasi-relational (Web) tables and spreadsheets tables.

Liu et al. [168] propose a learning-based semantic search algorithm to suggest appropriate Semantic Web terms and ontologies for the given data. The approach combines various measures for semantic similarity of documents to build a weighted feature-based semantic search model, which is then able to find the most suitable ontologies. The weights are learned from training data, using subgradient descent method and logistic regression.

Limaye et al. [166] propose a new probabilistic graphical model for simultaneously choosing entities for cells, types for columns and relations for column pairs, using YAGO as a background knowledge base. For building the graphical models, several types of features were used, i.e., cell text and entity label, column type and type label, column type and cell entity, relation and pair of column types, relation

and entity pairs. The experiments showed that approaching the three sub-problems collectively and in a unified graphical inference framework leads to higher accuracy compared to making local decisions.

Ventis et al. [317] associate multiple class labels (or concepts) with columns in a table and identify relations between the "subject" column and the rest of the columns in the table. Both the concept identification for columns and relation identification are based on maximum likelihood hypothesis, i.e., the best class label (or relation) is the one that maximizes the probability of the values given the class label (or relation) for the column. The evidences for the relations and for the classes are retrieved from a previously extracted isA database, describing the classes of the entities, and relations database, which contains relations between the entities. The experiments show that the approach can obtain meaningful labels for tables that rarely exist in the tables themselves, an that considering the recovered semantics leads to high precision search with little loss of recall of tables in comparison to document based approaches.

Wang et al. [321] propose a multi-phase algorithm that using the universal probabilistic taxonomy called *Probase* [328] is capable of understanding the entitles, attributes and values in many tables on the Web. The approach begins by identifying a single "entity column" in a table and, based on its values and rest of the column headers, associates a concept from the Probase knowledge base with the table.

Zhang et al. [339, 338] propose an incremental, bootstrapping approach that learns to label table columns using partial data in the column, and uses a generic feature model able to use various types of table context in learning. The work has been extended in [337], where the author shows that using sample selection techniques, it is possible to semantically annotate Web tables in a more efficient way.

Similarly, an approach for interpreting data from Web forms using LOD has been proposed [204]. The approach starts by extracting the attribute-value pars of the form, which is done using probing methods. Then, the data extracted from the Web forms are represented as RDF triple, or complete RDF graph. To enrich the graph with semantics, it is aligned with a large reference ontology, like YAGO, using ontology alignment approaches.

A particular case are tables in Wikipedia, which follow a certain structure and, with links to other Wikipedia pages, can be more easily linked to existing LOD sources such as DBpedia. Therefore, several approaches for interpreting tables from Wikipedia with LOD have been proposed. Munoz et al. [190, 184] propose methods for triplifying Wikipedia tables, called WikiTables, using existing LOD knowledge bases, like DBpedia and YAGO. Following the idea of the previous approaches, this approach starts by extracting entities from the tables, and then discovering existing relations between them. Similarly, a machine learning approach has been proposed by Bhagavatula et al. [15], where no LOD knowledge base is used, but only a meta data for the entities types and relations between them is added.

Similarly, approaches have been proposed for interpreting tabular data in spreadsheets [107, 156], CSV [71], and XML [111].

### 3.1.3 Using LOD to interpret unstructured data

Text mining is the process of analyzing unstructured information, usually contained in a natural language text, in order to discover new patterns. Most common text mining tasks include text categorization, text clustering, sentiment analysis and others. In most cases, text documents contain named entities that can be identified in real world, and further information can be extracted about them. Several approaches and APIs have been proposed for extracting named entities from text documents and linking them to LOD. One of the most used APIs is DBpedia Spotlight [174, 50], which allows for automatically annotating text documents with DBpedia URIs. This tool is used in several LOD enabled data mining approaches, e.g., [55, 273, 118, 269]. Several APIs for extracting semantic richness from text exist, like Alchemy API[8], OpenCalais API[9], Textwise SemanticHacker API[10]. All this APIs are able to annotate named entities with concepts from several knowledge bases, like DBpedia, YAGO, and Freebase. These tools and APIs have been evaluated in the NERD framework, implemented by Rizzo et al. [260].

Furthermore, Linked Open Data is also heavily used for better understanding of social media, which unlike authored news and other textual Web content, social media data pose a number of new challenges for semantic technologies, due to their large-scale, noisy, irregular, and social nature. An overview of tools and approaches for semantic representation of social media streams is given in [27]. This survey discusses five key research questions: (i) What ontologies and Web of Data resources can be used to represent and reason about the semantics of social media streams? For example, FOAF[11] and GUMO ontology [114] for describing people and social network, SIOC[12] and DLPO ontology [264] for modeling and interlinking social media, MOAT [217] ontology for modeling tag semantics (ii) How can semantic annotation methods capture the rich semantics implicit in social media? For example, keyphrase extraction [282, 241], ontology-based entity recognition, event detection [76] and sentiment detection citegangemi2014frame, sentilo. (iii) How can we extract reliable information from these noisy, dynamic content streams? (iv) How can we model users' digital identity and social media activities? For example, discovering user demographics [231], deriving user interests [2] and capturing user behavior [39]. (v) What semantic-based information access methods can help address the complex information seeking behavior in social media? For example, semantic search [1] and social media streams recommendation

[8] http://www.alchemyapi.com/api/

[9] http://www.opencalais.com/documentation/opencalais-documentation

[10] http://textwise.com/api

[11] http://xmlns.com/foaf/spec/

[12] http://sioc-project.org/

[41].

Once the user has developed a sufficient understanding of the domain, and the data mining task is defined, they need to select an appropriate data sample. If the data have already been mapped to appropriate domain specific ontologies or linked to external Linked Open Data, the users can more easily select a representative sample and/or meaningful subpopulation of the data for the given data mining task. For example, for a collection of texts, the user may decide to select those which mention a politician *after* the data has been linked to the semantic web, so that such a selection becomes possible.

Table 3.1 gives an overview of the discussed approaches in this section.[13] It can be observed that at the selection step, links between datasets play only a minor role, and reasoning is scarceley used. In most cases, general-purpose knowledge bases, such as DBpedia or YAGO, are used as sources of knowledge.

The selection of relevant semantic web datasets is usually done by *interlinking* a dataset at hand with data from Linked Open Data. There are strategies and tools for different kinds of data: relational databases are typically mapped to the semantic web using mapping rules and tools such as D2R. In those cases, mapping rules are typically written manually, which is easily possible because the schema of a relational database is usually explicitly defined.
Semi-structured data, such as Web tables, usually comes without explicit semantics, and in large quantities. Here, different heuristics and machine learning approaches are often applied to link them to LOD sources. For that case, it has been shown that combining approaches which perform schema and instance matching in a holistic way typically outperform approaches that handle both tasks in isolation.
For unstructured data, i.e., textual contents, the interlinking is typically done by linking named entities in the text to LOD sources with tools such as DBpedia Spotlight.
Once the interlinking is done, data visualization and summarization techniques can benefit from the additional knowledge contained in the interlinked datasets.

[13] The tables used for summarizing approaches at the end of each section are structured as follows: The second column of the table states the problem domain on which the approach is applied. The third column states the data mining task/domain that was used in the approach. The next two columns capture the characteristics of the ontologies used in the approach, i.e., the complexity level of the ontology, and if reasoning is applied on the ontology. Based on a prior categorization of ontologies presented in [228], we distinguish two degrees of ontology complexity: ontologies of low complexity that consist of class hierarchies and subclass relations (marked with *L*), and onotologies with high complexity that also contain relations other than the subclass relations, and further constraints, rules and so on (marked with *H*). The sixth column indicates if links (such as *owl:sameAs*) to other LOD sources were followed to extract additional information. The next column states whether explicit semantic information were used from a given LOD source. The final two columns list the used LOD sources and shared ontologies, respectively. If a LOD source is used, the respective ontology is used as well, without explicitly stating that in the table.

Table 3.1: Summary of approaches used in the selection step.

| Approach | Domain | | Ontology | | LOD | | Used Datasets | |
|---|---|---|---|---|---|---|---|---|
| | Problem | Data Mining | Complexity | Reasoning | Links | Semantics | LOD | Ontology |
| [189, 188, 294, 185, 186, 187] | persons, places, organizations | / | H | yes | no | yes | DBpedia, YAGO, Freebase, WordNet | Wikitology |
| [107] | biology | / | L | no | no | no | / | / |
| [156] | commerce | / | H | yes | no | no | DBpedia | / |
| [111] | medicine, publications | / | H | no | yes | yes | ClinicTrials.gov, BibBase[a] | / |
| [166] | persons, places, organizations | / | H | no | no | yes | DBpedia, YAGO, WordNet | / |
| [317] | geography | / | H | no | no | no | YAGO, Freebase | / |
| [321] | persons, places, organizations | / | H | no | yes | yes | Probase, DBpedia | / |
| [339, 338, 337] | persons, places, organizations, music, movies | / | H | yes | yes | yes | DBpedia, YAGO, MusicBrainz[b] | / |
| [204] | books | / | H | no | no | yes | YAGO | / |

## 3.2 Preprocessing

Once the data is mapped to domain specific knowledge, the constraints expressed in the ontologies can be used to perform data validity checks and data cleaning. Ontologies can be used for detecting outliers and noise, as well as for handling missing values and data range and constraint violations, and guiding the users through custom preprocessing steps.

Ontologies are often used in many research approaches for the use of data cleaning and data preprocessing. Namely, there are two applications of ontologies in this stage: domain-independent ontologies used for data quality management, and domain ontologies. The first category of ontologies usually contain specifications for performing cleaning and preprocessing operations. In these approaches, the ontology is usually used to guide the user through the process of data cleaning and validation, by suggesting possible operations to be executed over the data. The second category of ontologies provide domain specific knowledge needed to validate and clean data, usually in an automatic manner.

### 3.2.1 Domain-independent Approaches

One of the first approaches that uses a data quality ontology is proposed by Wang et al. [323]. They propose a framework called *OntoClean*[14] for ontology-based data cleaning. The core component of the framework is the data cleaning ontology component, which is used when identifying the cleaning problem and the relevant data. Within this component, the task ontology specifies the potential methods that may be suitable for meeting the user's goals, and the domain ontology includes all classes, instances, and axioms in a specific domain, which provides domain knowledge such as attribute constraints for checking invalid values during performing the cleaning tasks.

A similar approach is proposed by Perez et al. [240] with the *OntoDataClean* framework, which is able to guide the data cleaning process in a distributed environment. The framework uses a preprocessing ontology to store the information about the required transformations. First, the process of identifying and storing the required preprocessing steps has to be carried by a domain expert. Then, these transformations are needed to homogenize and integrate the records so they can be correctly analyzed or unified with other sources. Finally, the required information are stored in the preprocessing ontology, and the data transformations can be accomplished automatically. The approach has been tested on four databases in the domain of bio-medicine, showing that using the ontology the data can be correctly preprocessed and transformed according the needs.

[14] Not to be confused with the ontology engineering method by Guarino and Welty.

### 3.2.2 Domain-specific Approaches

One of the first approaches to use a domain specific ontology is proposed by Philips et al. [234]. The approach uses ontologies to organize and represent knowledge about attributes and their constrains from relational databases. The approach is able to automatically, or semi-automatically with an assist of the user, identify the domains of the attributes, relations between the attributes, duplicate attributes and duplicate entries in the database.

Kedad et al. [143] propose a method for dealing with semantic heterogeneity during the process of data cleaning when integrating data from multiple sources, which is differences in terminologies. The proposed solution is based on linguistic knowledge provided by a domain is-a ontology. The main idea is to automatically generate correspondence assertions between instances of objects based on the is-a hierarchy, where the user can specify the level of accuracy expressed using the domain ontology. Once the user has specified the level of accuracy, two concepts will be considered the same if there is a subsumption relation between them, or both belong to the same class. Using this approach the number of results might be increased when querying the data, e.g., for the query "Do red cars have more accidents than others?" the system will not only look for *red cars*, but also for cars with color *ruby*, *vermilion*, and *seville*, which are subclasses of the red color.

Milano et al. introduce the OXC framework [179] that allows data cleaning on XML documents based on a uniform representation of domain knowledge through an ontology, which is gathered from domain analysis activities and from the DTDs of the documents. The framework comprises a methodology for data quality assessment and cleaning based on the reference ontology, and an architecture for XML data cleaning based on such methodology. Given a domain ontology, a mapping relation between the DTD and the ontology is defined, which is used to define quality dimensions (accuracy, completeness, consistency and currency), and perform data quality improvement by relying on the semantics encoded by the ontology.

Brueggemann et al. [31] propose a combination of domain specific ontologies and data quality management ontologies, by annotating domain ontologies with data quality management specific metadata. The authors have shown that such hybrid approach is suitable for consistency checking, duplicate detection, and metadata management. The approach has been extended in [30], where correction suggestions are being generated for each detected inconsistency. The approach uses the hierarchical structure of the ontology to offer the user semantically related context-aware correction suggestions. Moreover, the framework uses several measurements of semantic distances in ontologies to find the most suitable corrections for the identified inconsistencies. Based on those metrics the system can offer several suggestions for value corrections, i.e., value of next-sibling, first-child and parent. The approach has been applied on data from the cancer registry of Lower Saxony[15], showing that it can successfully support domain experts.

[15] `http://www.krebsregister-niedersachsen.de`

Wang et al. [324] present a density-based outlier detecting method using domain ontology, named *ODSDDO* (Outlier Detecting for Short Documents using Domain Ontology). The algorithm is based on the *local outlier factor* algorithm, and uses domain ontology to calculate the semantic distance between short documents which improves the outlier detecting precision. To calculate the semantic similarity between two documents, first each word from each document is mapped to the corresponding concept in the ontology. Then, using the ontology concept tree, the similarity between each pair of concepts is calculated. The distance between two documents is then simply calculated as average of the sum of the maximum similarities between the pairs of concepts. The documents that have small or zero semantic similarity to other documents in the dataset are considered to be outliers.

Lukaszewski [172] propose an approach to admit and utilize noisy data by enabling to model different levels of knowledge granularity both in training and testing examples. The authors argue that erroneous or missing attribute values may be introduced by users of a system that are required to provide very specific values, but the level of their knowledge of the domain is too general to precisely describe the observation by the appropriate value of an attribute. Therefore, they propose knowledge representation that uses hierarchies of sets of attribute values, derived from subsumption hierarchies of concepts from an ontology, which decreases the level of attribute-noise in the data.

Füber and Hepp [90, 89, 91, 92] propose approaches for using Semantic Web technologies and Linked Open Data to reduce the effort for data quality management in relational databases. They show that using LOD reference data can help identifying missing values, illegal values, and functional dependency violations. In their first work [90], the authors describe how to identify and classify data quality problems in relational databases, through the use of SPARQL Inferencing Notation (SPIN)[16]. SPIN is a Semantic Web vocabulary and processing framework that facilitates the representation of rules based on the syntax of the SPARQL protocol and RDF query language. To apply the approach on relational databases, the D2RQ tool [19] is used to extract data from relational databases into an RDF representation. The framework allows domain experts to define data requirements for their data based on forms as part of the data quality management process. The SPIN framework then automatically identifies requirement violations in data instances, i.e. syntactic errors, missing values, unique values violations, out or range values, and functional dependency violations. This approach is extended in [91] to asses the quality state of data in additional dimensions.

In a further work [89], instead of manually defining the data validation rules, the authors propose using Linked Open Data as trusted knowledge base that already contains information on the data dependencies. This approach have been shown to significantly reduce the effort for data quality management, when reference data is available in the LOD cloud. The approach was evaluated against a local knowledge

[16] `http://spinrdf.org/`

base that contained manually created address data. Using GeoNames as a reference LOD dataset, the approach was able to identify invalid city entries, and invalid city-country relations.

A similar approach using SPIN, has been developed by Moss et al. [183] for assessing medical data. The system compromises a set of ontologies that support reasoning in a medical domain, such as human psychology, medical domain, and patient data. To perform the data cleaning, several rules for checking missing data points and value checking were used. The approach is evaluated on data from the Brain-IT network[17], showing that it is able to identify invalid values in the data. Ontologies are often used in the healthcare domain for data quality management and data cleaning. Literature review of such papers is presented in [165].

In [163] we have developed an approach for filling missing values in a local table using LOD, which is implemented in a system named *Mannheim Search Joins Engine*[18]. The system relies on a large data corpus, crawled from over one million different websites. Besides two large quasi-relational datasets, the data corpus includes the *Billion Triples Cahllenge 2014 Dataset*[19] [136], and the *WebData-Commons Microdata Dataset*[20] [175]. For a given local table, the engine searches the data corpus for additional data for the attributes of the entities in the input table. To perform the search, the engine uses the existing information in the table, i.e. the entities' labels, the attributes' headers, and the attributes' data types. The discovered data is usually retrieved from multiple sources, therefore the new data is first consolidated using schema matching and data fusion methods. Then, the discovered data is used to fill the missing values in the local table. Also, the same approach can be used for validating the existing data in the given table i.e. outlier detection, noise detection and correction.

Table 3.2 gives an overview of the discussed approaches in this section. We can observe that, while ontologies are frequently used for data cleaning, well-known LOD datasets like DBpedia are scarcely exploited. Furthermore, many approaches have been tailored to and evaluated in the medical domain, likely because quite a few sophisticated ontologies exist in that domain.

Ontologies and Semantic Web data help with preprocessing the data, mostly for increasing the data quality. There are various data quality dimensions that can be addressed. Outliers and false values may be found by identifying data points and values that violate constraints defined in those ontologies. Subsumption hierarchies and semantic relations help unifying synonyms and detecting interrelations between attributes. Finally, missing values can be inferred and/or filled from LOD datasets.

[17] http://www.brain-it.eu/
[18] http://searchjoins.webdatacommons.org/
[19] http://km.aifb.kit.edu/projects/btc-2014/
[20] http://webdatacommons.org/structureddata/

Table 3.2: Summary of approaches used in the preprocessing step.

| Approach | Domain | | Ontology | | LOD | | Used Datasets | |
|---|---|---|---|---|---|---|---|---|
| | Problem | Data Mining | Complexity | Reasoning | Links | Semantics | LOD | Ontology |
| [323] | geography | / | H | no | no | no | / | OntoClean ontology |
| [240] | biomedicine | / | H | no | no | no | / | OntoDataClean ontology |
| [143] | medicine | / | H | no | no | no | / | custom ontology |
| [143] | medicine | / | H | no | no | no | / | custom ontology |
| [179] | / | / | H | no | no | no | / | custom ontology |
| [31, 30] | medicine | / | H | yes | no | no | / | custom ontology |
| [324] | social media | outlier detection | H | no | no | no | / | custom ontology |
| [90, 89, 91, 92] | geography | / | H | yes | no | yes | DBpedia, GeoNames[a] | / |
| [163] | geography, companies, movies, books, music, persons, drugs | / | H | no | yes | yes | BTC 2014, WebData-Commons Microdata Dataset | / |
| [183] | medicine | / | H | no | no | no | / | custom ontology |

## 3.3 Transformation

At this stage, the generation of better data for the data mining process is prepared. The transformation step includes dimensionality reduction, feature generation and feature selection, instance sampling, and attribute transformation, such as discretization of numerical data, aggregation, functional transformations, etc. In the context of Semantic Web enabled data mining, *feature generation* and *feature selection* are particularly relevant.

### 3.3.1 Feature Generation

Linked Open Data has been recognized as a valuable source of background knowledge in many data mining tasks. Augmenting a dataset with features taken from Linked Open Data can, in many cases, improve the results of a data mining problem at hand, while externalizing the cost of creating and maintaining that background knowledge [221].

Most data mining algorithms work with a propositional *feature vector* representation of the data, i.e., each instance is represented as a vector of features $\langle f_1, f_2, ..., f_n \rangle$, where the features are either binary (i.e., $f_i \in \{true, false\}$), numerical (i.e., $f_i \in \mathbb{R}$), or nominal (i.e., $f_i \in S$, where $S$ is a finite set of symbols) [211]. Linked Open Data, however, comes in the form of *graphs*, connecting resources with types and relations, backed by a schema or ontology.

Thus, for accessing Linked Open Data with existing data mining tools, transformations have to be performed, which create propositional features from the graphs in Linked Open Data, i.e., a process called *propositionalization* [154]. Usually, binary features (e.g., `true` if a type or relation exists, `false` otherwise) or numerical features (e.g., counting the number of relations of a certain type) are used. Furthermore, elementary numerical or nominal features (such as the population of a city or the production studio of a movie) can be added [225]. Other variants, e.g., computing the fraction of relations of a certain type, are possible, but rarely used.

In the recent past, a few approaches for propositionalizing Linked Open Data for data mining purposes have been proposed. Many of those approaches are supervised, i.e., they let the user formulate SPARQL queries, which means that they leave the propositionalization strategy up to the user, and a fully automatic feature generation is not possible. Usually, the resulting features are binary, or numerical aggregates using `SPARQL COUNT` constructs.

*LiDDM* [140] is an integrated system for data mining on the Semantic Web. The tool allows the users to declare SPARQL queries for retrieving features from LOD that can be used in different machine learning techniques, such as clustering and classification. Furthermore the tool offers operators for integrating data from multiple sources, data filtering and data segmentation, which are carried manually by the user. The usefulness of the tool has been presented through two use

cases, using DBpedia, World FactBook[21] and LinkedMDB[22], in the application of correlations analysis and rule learning.

A similar approach has been used in the RapidMiner[23] semweb plugin [145], which preprocesses RDF data in a way that it can be further processed by a data mining tool, RapidMiner in that case. Again, the user has to specify a SPARQL query to select the data of interest, which is then converted into feature vectors. The authors propose two methods for handling set-values data, by mapping them into an N-dimensional vector space. The first one is *FastMap*, which embeds points in an N-dimensional space based on a distance metric, much like Multidimensional Scaling (MDS). The second one is Correspondence Analysis (CA), which maps values to a new space based on their cooccurrence with values of other attributes. The approaches were evaluated on IMDB data[24], showing that the mapping functions can improve the results over the baseline.

Cheng et al. [43] propose an approach for automated feature generation after the user has specified the type of features. To do so, the users have to specify the SPARQL query, which makes this approach supervised. The approach has been evaluated in the domain of recommender systems (movies domain) and text classification (tweets classification). The results show that using semantic features can improve the results of the learning models compared to using only standard features.

Mynarz et al. [192] have considered using user specified SPARQL queries in combination with SPARQL aggregates, including `COUNT`, `SUM`, `MIN`, `MAX`. Kauppinen et al. have developed the SPARQL package for R[25] [141, 142], which allows importing LOD data in the very well known environment for statistical computing and graphics *R*. In their research they use the tool to perform statistical analysis and visualization of the linked Brazilian Amazon rainforest data. The same tool has been used in [312] for statistical analysis in piracy attack reports data. Moreover, they use the tool to import RDF data from multiple LOD sources in the environment of *R*, which allows them to easier analyze, interpret and visualize the discovered patterns in the data.

*FeGeLOD* [225] was the first fully automatic approach for enriching data with features that are derived from LOD. In that work, the authors have proposed six different feature generation strategies, allowing for both binary features and simple numerical aggregates. The first two strategies are only concerned with the instances themselves, i.e., retrieving the data properties of each entity, and the types of the entity. The four other strategies consider the relation of the instances to other resources in the graph, i.e. incoming and outgoing relations, and *qualified relations*, i.e., aggregates over the type of both the relation and the related entity.

The related work indicates that there is a gap between Semantic Web knowl-

[21] `ttp://wifo5-03.informatik.uni-mannheim.de/factbook/`
[22] `http://www.linkedmdb.org/`
[23] `http://www.rapidminer.com/`
[24] `http://www.imdb.com/`
[25] `http://linkedscience.org/tools/sparql-package-for-r/`

edge graphs and existing data mining tools. More precisely, there is a lack of tools and approaches for accessing background knowledge from Semantic Web knowledge graphs for data minining. This leads to formulating the research question RQ2 in Section 1.1. In Chapter 7 we present the RapidMiner Linked Open Data extension, which addresses this question. Currently, the RapidMiner LOD extension supports the user in all steps of the LOD enabled knowledge discovery process. i.e. linking, combining data from multiple LOD sources, preprocessing and cleaning, transformation, data analysis, and interpretation of data mining findings.

A problem similar to feature generation is addressed by *Kernel functions*, which compute the distance between two data instances. The similarity is calculated by counting common substructures in the graphs of the instances, e.g., walks, paths and threes. The graph kernels are used in kernel-based data mining and machine learning algorithms, most commonly support vector machines (SVMs), but can also be exploited for tasks such as clustering. In the past, many graph kernels have been proposed that are tailored towards specific application [128, 127, 126], or towards specific semantic representation [80, 81, 23, 16]. But only a few approaches are general enough to be applied on any given RDF data, regardless of the data mining task. Lösch et al. [169] introduces two general RDF graph kernels, based on intersection graphs and intersection trees. First, they propose the use of walk and path kernels, which count the number of walks and paths in the intersected graphs. Then, they propose full subtree kernel, which counts the number of full sub-trees of the intersection tree.

The intersection tree path kernel introduced by Lösch et al., has been modified and simplified by Vries et al. [58, 57, 24, 59], which also allows for explicit calculation of the instances' feature vectors, instead of pairwise similarities. Computing the feature vectors significantly improves the computation time, and allows using any arbitrary machine learning methods. They have developed two types of kernels over RDF data, RDF walk count kernel and RDF WL sub tree kernel. The RDF walk count kernel counts the different walks in the sub-graphs (up to the provided graph depth) around the instances nodes. The RDF WL sub tree kernel counts the different full sub-trees in the sub-graphs (up to the provided graph depth) around the instances nodes, using the Weisfeiler-Lehman algorithm [277]. The approaches developed by Lösch et al. and by Vries et al. have been evaluated on two common relational learning tasks: entity classification and link prediction.

In Chapter 5 and Chapter 7, we describe several approaches for feature generation from Semantic Web knowledge graphs.

Another line of work for generating data mining features from Semantic Web knowledge graphs, are graph embeddings. In Chapter 8 and Chapter 9, we describe an approach for embedding complete Semantic Web knowledge graphs. Generally, our work is closely related to the approaches DeepWalk [233] and Deep Graph Kernels [329]. DeepWalk uses language modeling approaches to learn social representations of vertices of graphs by modeling short random-walks on large social graphs, like BlogCatalog, Flickr, and YouTube. The Deep Graph Kernel approach extends the DeepWalk approach by modeling graph substructures, like graphlets,

instead of random walks. Node2vec [101] is another approach very similar to DeepWalk, which uses second order random walks to preserve the network neighborhood of the nodes. The approach we propose in this chapter differs from these approaches in several aspects. First, we adapt the language modeling approaches on directed labeled RDF graphs, unlike the approaches mentioned above, which work on undirected graphs. Second, we show that task-independent entity vectors can be generated on large-scale knowledge graphs, which later can be reused on a variety of machine learning tasks on different datasets.

Furthermore, multiple approaches for knowledge graph embeddings for the task of link prediction have been proposed [196, 322], which could also be considered as approaches for generating propositional features from graphs. RESCAL [198] is one of the earliest approaches, which is based on factorization of a three-way tensor. The approach is later extended into Neural Tensor Networks (NTN) [281] which can be used for the same purpose. One of the most successful approaches is the model based on translating embeddings, TransE [28]. This model builds entity and relation embeddings by regarding a relation as translation from head entity to tail entity. This approach assumes that some relationships between words could be computed by their vector difference in the embedding space. However, this approach cannot deal with reflexive, one-to-many, many-to-one, and many-to-many relations. This problem was resolved in the TransH model [325], which models a relation as a hyperplane together with a translation operation on it. More precisely, each relation is characterized by two vectors, the norm vector of the hyperplane, and the translation vector on the hyperplane. While both TransE and TransH, embed the relations and the entities in the same semantic space, the TransR model [167] builds entity and relation embeddings in separate entity space and multiple relation spaces. This approach is able to model entities that have multiple aspects, and various relations that focus on different aspects of entities. While such approaches have been used for the task of link prediction, they haven't been considered for data mining tasks. This leads to formulating research question RQ3 in Section 1.1. We address this question in Chapter 8 and Chapter 9, where we introduce an approach for embedding Semantic Web knowledge graphs, which we show to outperform the related work regarding complexity and performance in various data mining tasks.

### 3.3.2 Feature Selection

We have shown that there are several approaches that generate propositional feature vectors from Linked Open Data. Often, the resulting feature spaces can have a very high dimensionality, which leads to problems both with respect to the performance as well as the accuracy of learning algorithms. Thus, it is necessary to apply some feature selection approaches to reduce the feature space. Additionally, for datasets that already have a high dimensionality, background knowledge from LOD or lingustic resources such as WordNet may help reducing the feature space better than standard techniques which do not exploit such background knowledge.

Feature selection is a very important and well studied problem in the literature. The objective is to identify features that are correlated with or predictive of the class label. Generally, all feature selection methods can be divided into two broader categories: wrapper methods and filter methods (John et al. [134] and Blum et al. [25]). The wrapper methods use the predictive accuracy of a predetermined learning method to evaluate the relevance of the feature sub set. Because of their large computational complexity, the wrapper methods are not suitable to be used for large feature spaces. Filter methods are trying to select the most representative sub-set of features based on a criterion used to score the relevance of the features. In the literature several techniques for scoring the relevance of features exist, e.g., Information Gain, $\chi^2$ measure, Gini Index, and Odds Ratio.

However, standard feature selection methods tend to select the features that have the highest relevance score without exploiting the hierarchical structure of the feature space. Therefore, using such methods on hierarchical feature spaces, may lead to the selection of redundant features, i.e., nodes that are closely connected in the hierarchy and carry similar semantic information.

While there are a lot of state-of-the-art approaches for feature selection in standard feature space [182], only few approaches for feature selection in hierarchical feature space are proposed in the literature.

In feature vectors generated from external knowledge we can often observe relations between the features. In many cases those relations are hierarchical relations, or we can say that the features subsume each other, and carry similar semantic information. Those hierarchical relations can be easily retrieved from the ontology or schema used for publishing the LOD, and can be used to perform better feature selection.

Jeong et al. [133] propose the *TSEL* method using a semantic hierarchy of features based on WordNet relations. The presented algorithm tries to find the most representative and most effective features from the complete feature space. To do so, they select one representative feature from each path in the tree, where path is the set of nodes between each leaf node and the root, based on the *lift* measure, and use $\chi^2$ to select the most effective features from the reduced feature space.

Wang et al. [320] propose a *bottom-up hill climbing* search algorithm to find an optimal subset of concepts for document representation. For each feature in the initial feature space, they use a kNN classifier to detect the $k$ nearest neighbors of each instance in the training dataset, and then use the purity of those instances to assign scores to features.

Lu et al. [171] describe a *greedy top-down* search strategy for feature selection in a hierarchical feature space. The algorithm starts with defining all possible paths from each leaf node to the root node of the hierarchy. The nodes of each path are sorted in descending order based on the nodes' information gain ratio. Then, a greedy-based strategy is used to prune the sorted lists. Specifically, it iteratively removes the first element in the list and adds it to the list of selected features. Then, removes all ascendants and descendants of this element in the sorted list. Therefore, the selected features list can be interpreted as a mixture of concepts

from different levels of the hierarchy.

In Chapter 6, we introduce an approach [253] that exploits hierarchies for feature selection in combination with standard metrics, such as *information gain* or *correlation*

Furthemore, when creating features from multiple LOD sources, often a single semantic feature can be found in multiple LOD source represented with different properties. For example, the area of a country in DBpedia is represented with *db:areaTotal*, and with *yago:hasArea* in YAGO. The problem of aligning properties, as well as instances and classes, in ontologies is addressed by *ontology matching* techniques [79]. Even though there exist a vast amount of work in the area of ontology matching, most of the approaches for generating features from Linked Open Data are not explicitly addressing this problem. In Chapter 7, we describe an approach that is able to match properties extracted from multiple LOD sources, which are later fused into a single feature.

In pattern mining and association rule mining, domain ontologies are often used to reduce the feature space in order to get more meaningful and interesting patterns. In the approach proposed by Bellandi et al. [10] several domain-specific and user-defined constraints are used, i.e., pruning constraints, used to filter uninteresting items, and abstraction constraints permitting the generalization of items toward ontology concepts. The data is first preprocessed according to the constraints extracted from the ontology, and then, the data mining step takes place. Applying the pruning constraints excludes the information that the user is not interested in, before applying the data mining approach.

Onto4AR is a constraint-based algorithm for association mining proposed by Antunes [6] and revised later in [7], where taxonomical and non-taxonomical constraints are defined over an item ontology. This approach is interesting in the way that the ontology offers a high level of expression for the constraints, which allows to perform the knowledge discovery at the optimal level of abstraction, without the need for user input. Garcia et al. developed a technique called *Knowledge Cohesion* [97, 17] to extract more meaningful association rules. The proposed metric is based on semantic distance, which measures how close two items are semantically based within the ontology, where each type of relation is weighted differently.

### 3.3.3 Other

Zeman et al. [334] present the Ferda DataMiner tool, which is focused on the data transformation step. In this approach the ontologies are used for two purposes: construction of adequate attribute categorization, and identification and exploitation of semantically related attributes. The authors claim that ontologies can be efficiently used for categorization of attributes as higher-level semantics could be assigned to individual values. For example, for blood pressure there are predefined values that divide the domain in a meaningful way: say, blood pressure above 140/90 mm Hg is considered as hypertension. For the second purpose, ontologies are used to discover the relatedness between the attributes, which can be exploited so as to

meaningfully arrange the corresponding data attributes in the data transformation phase.

Table 3.3 gives an overview of the discussed approaches in this section. It can be observed that at this stage of the data mining process, many approaches also exploit links between LOD datasets to identify more features. On the other hand, the features are most often generated without regarding the schema of the data, which is, in most cases, rather used for post processing of the features, e.g., for feature selection. Likewise, reasoning is only scarcely used.

Most data mining algorithms and tools require a *propositional* representation, i.e., feature vectors for instances. Typical approaches for propositionalization are, e.g., adding all numerical datatype properties as numerical features, or adding all direct types as binary features. There are unsupervised and supervised methods, where for the latter, the user specifies a query for features to generate – those are useful if the user knows the LOD dataset at hand and/or has an idea which features could be valuable. While such classic propositionalization methods create human interpretable features and thus are also applicable for *descriptive* data mining, kernel methods often deliver better *predictive* results, but at the price of losing the interpretability of those results.
A crucial problem when creating explicit features from Linked Open Data is the scalability and the number of features generated. Since only few approaches focus on identifying high value features already at the generation step, combining feature generation with feature subset selection is clearly advised.
The schema information for the LOD sources, such as type hierarchies, can be exploited for feature space reduction. There are a few algorithms exploiting the schema, which often provide a better trade-off between feature space reduction and predictive performance than schema-agnostic approaches.

## 3.4 Data Mining

After the data is selected, preprocessed and transformed in the most suitable representation, the next step is choosing the appropriate data mining task and data mining algorithm. Depending on the KDD goals, and the previous steps of the process, the users need to decide which type of data mining to use, i.e. classification, regression, clustering, summarization, or outlier detection. Understanding the domain will assist in determining what kind of information is needed from the KDD process, which makes it easier for the users to make a decision. There are two broader categories of goals in data mining: prediction and description. Prediction is often referred to as supervised data mining, which attempts to forecast the possible future or unknown values of data elements. On the other hand, descriptive data mining is referred as unsupervised data mining, which seeks to discover interpretable patterns in the data. After the strategy is selected, the most appropriate data mining algorithm should be selected. This step includes selecting methods to

Table 3.3: Summary of approaches used in the transformation step.

| Approach | Domain | | Ontology | | LOD | | Used Datasets | |
|---|---|---|---|---|---|---|---|---|
| | Problem | Data Mining | Complexity | Reasoning | Links | Semantics | LOD | Ontology |
| [140] | government, economy, movies | correlation, association mining | H | no | yes | no | DBpedia, World Fact-Book[a], LinkedMDB | / |
| [145] | movies | classification | H | no | yes | no | DBpedia, LinkedMDB | / |
| [43] | movies, social media | recommender systems, classification | H | no | yes | no | YAGO | / |
| [141, 142] | geography | correlations | L | no | yes | yes | Linked Brazilian Amazon Rainforest, DBpedia | / |
| [225] | biology, sociology, economy | classification, regression | H | no | yes | yes | DBpedia | / |
| [246] | economy, publications | correlations | H | no | yes | yes | DBpedia, YAGO, LinkedGeoData[b], Eurostat[c], GeoNames, WHO[d], Linked Energy Data[e], OpenCyc[f], World Factbook | / |

| Approach | Domain | | Ontology | | LOD | | Used Datasets | |
|---|---|---|---|---|---|---|---|---|
| | Problem | Data Mining | Complexity | Reasoning | Links | Semantics | LOD | Ontology |
| [55] | news | sentiment analysis | H | no | yes | no | DBpedia | / |
| [222] | music, movies, books | linkage error detection | H | no | yes | yes | DBpedia, DBTropes[a], Peel Sessions [b] | / |
| [58, 57] | publications, geology | property value prediction, link prediction | H | no | no | no | custom dataset, British Geological Survey[c] | SWRC[d] |
| [24] | bio-medicine, publications | classification | H | no | yes | yes | MUTAG, ENZYMES, Semantic Web Conference Corpus[e], British Geological Survey | / |
| [133] | news | text classification | H | no | no | no | WordNet | / |
| [320] | biomedicine | text classification | H | no | no | no | / | UMLS |
| [171] | pharmacology | classication | H | no | no | no | / | NDF-RT[f] |
| [10] | commerce | rule learning | H | no | no | no | / | products ontology |
| [6, 7] | movies | association rules | H | no | no | no | / | custom ontology |
| [334] | medicine | association mining | H | yes | no | no | / | UMLS[g] |
| [97, 17] | accident reports | text mining, rule learning | H | no | no | no | / | custom ontology |

search patterns in the data, and deciding on specific models and parameters of the methods.

Once the data mining method and algorithm are selected, the data mining takes place.

To the best of our knowledge, there are rarely any approaches in the literature that incorporates data published as Linked Open Data into the data mining algorithms themselves. However, many approaches are using ontologies for the data mining process, not to only support the user in the stage of selecting the data mining methods, but to guide the users through the whole process of knowledge discovery.

### 3.4.1 Domain-independent Approaches

While there is no universally established data mining ontology yet, there are several data mining ontologies currently under development, such as the Knowledge Discovery (KD) Ontology [332], the KDDONTO Ontology [69], the Data Mining Workflow (DMWF) Ontology[26][146], the Data Mining Optimization (DMOP) Ontology[27] by Hilario [119, 120], OntoDM[28] [212, 213], and its sub ontology modules OntoDT[29], OntoDM-core[30][215] and OntoDM-KDD[31] [214].

An overview of existing intelligent assistants for data analysis that use ontologies is given in [276]. In this survey, all approaches are categorized by several criteria. First, which types of support the intelligent assistants offer to the data analyst. Second, it surveys the kinds of background knowledge that the IDAs rely on in order to provide the support. Finally, performs thorough comparison of IDAs in light of the defined dimensions and the identification of limitations and missing features.

One of the earliest approaches, *CAMLET*, was proposed by Suyama et al. [292], which uses two light-weight ontologies of machine learning entities to support the automatic composition of inductive learning systems.

Among the first prototypes is the *Intelligent Discovery Assistant* proposed by Bernstein et al. [14], which provides users with systematic enumerations of valid sequences of data mining operators. The tool is able to determine the characteristics of the data and of the desired mining result, and uses an ontology to search for and enumerate the KDD processes that are valid for producing the desired result from the given data. Also, the tool assists the user in selecting the processes to execute, by ranking the processes according to what is important to the user. A light-weight ontology is used that contains only a hierarchy of data mining operators divided into three main classes: preprocessing operators, induction algorithms

[26] http://www.e-lico.eu/dmwf.html
[27] http://www.e-lico.eu/DMOP.html
[28] http://www.ontodm.com/doku.php
[29] http://www.ontodm.com/doku.php?id=ontodt
[30] http://www.ontodm.com/doku.php?id=ontodm-core
[31] http://www.ontodm.com/doku.php?id=ontodm-kdd

and post processing operators.

Many approaches are using Semantic Web technologies to assist the user in building complex data mining workflows. Žáková et al. [331, 332] proposed an approach for semiautomatic workflow generation that requires only the user input and the user desired output to generate complete data mining workflows. To implement the approach, the authors have developed the knowledge discovery ontology, which gives a formal representation of a knowledge types and data mining algorithms. Second, a planning algorithm is implemented that assembles workflows based on the planning task descriptions extracted from the knowledge discovery ontology and the given user's input-output task requirements. In such semiautomatic environment the user is not required to be aware of the numerous properties of the wide range of relevant data mining algorithms. In their later work, the methodology is implemented in the Orange4WS environment for service-oriented data mining [333, 238].

Diamantini et al. [68] introduce a semantic based, service-oriented framework for tools sharing and reuse, giving advanced support for the semantic enrichment through semantic annotation of KDD tools, deployment of the tools as web services and discovery and use of such services. To support the system an ontology named KDDONTO [69] is used, which represents a formal ontology describing the domain of KDD algorithms. The ontology provides information required by the KDD composer to assist them to choose the suitable algorithms for achieving their goal starting from the data at hand, and to correctly compose the algorithms for forming a valid process [70].

Kietz et al. [146, 148] presented a data mining ontology for workflow planning, able to effectively organize hundreds of operators, which is the base for checking the correctness of KDD workflows and an Hierarchical Task Network planning component able to effectively enumerate useful KDD workflows. This includes the objects manipulated by the system, the meta data needed, the operators used, and a goal description. The workflow generator is tightly coupled with a meta-miner whose role is to rank the workflows and is based on the DMOP ontology. Furthermore, the authors introduced the eProPlan tool [147], which represents ontology-based environment for planning KDD workflows. Later on, the tool is used to semantically annotate all operators in the very well known data mining tool RapidMiner. This allows the users to easily, and faster build more efficient KDD workflows within RapidMiner [149]. Their evaluation showed that using Semantic Web technologies can speed up the workflow design time up to 80%. This is achieved by automatic suggestion for possible operations in all phases of the KDD process.

Furthermore, Hilario et al. [119] present the data mining optimization ontology, which provides a unified conceptual framework for analyzing data mining tasks, algorithms, models, datasets, workflows and performance metrics, as well as their relationships. One of the main goals of the ontology is to support meta-mining of complete data mining experiments in order to extract workflow patterns [120]. In addition, the authors have developed a knowledge base by defining instances of

the DMOP ontology. The DMOP ontology is not based on any upper-level ontology and uses a large set of customized special-purpose relations.

Panov et al. [212, 213] propose an ontology of data mining *OntoDM* that includes formal definitions of basic data mining entities, such as *datatype* and *dataset*, *data mining task* and *data mining algorithm*, which is based on the proposal for a general framework for data mining proposed by Džeroski [74]. The ontology is one of the first deep/heavy-weight ontology for data mining. To allow the representation of mining structured data, the authors developed a separate ontology module, named OntoDT, for representing the knowledge about datatypes. To represent core data mining entities, and to be general enough to represent the mining of structured data, the authors introduced the second ontology module called OntoDM-core [215]. The third, and final, module of the ontology is the OntoDM-KDD which is used for representing data mining investigations[214].

Gabriel et al. [94] propose the usage of semantic information about the attributes contained in a dataset for learning classification rules that are potentially better understandable. They use *semantic coherence*, i.e., the semantic proximity of attributes used in a rule, as a target criterion to increase the understandability of a rule. In their paper, they show that using WordNet as a source of knowledge, and adapting a standard separate-and-conquer rule learning algorithm [93], they can significantly increase the semantic coherence in a rule without a decrease in classification accuracy.

### 3.4.2 Domain-specific Approaches

Santos et al. [235] describes a research of an ontological approach for leveraging the semantic content of ontologies to improve knowledge discovery in databases. The authors divide the KDD process in three main operation, and try to support each of them using ontologies. First, in the data understanding and data preparation phases, ontologies can facilitate the integration of heterogeneous data and guide the selection of relevant data to be mined, regarding domain objectives. Second, during the modeling phase, domain knowledge allows the specification of constraints for guiding data mining algorithms by narrowing the search space. Finally, in the interpretation and evaluation phase, domain knowledge helps experts to validate and rank the extracted patterns.

Ding et al. [210, 209] introduce another ontology based framework for incorporating domain knowledge into data mining process. The framework is able to support the data mining process in several steps of the pipeline: data exploration, defining mining goals, data selection, data preprocessing and feature selection, data transofmation, data mining algorithm parameter selection, and data mining results evaluation.

Češpivová et al. [38] have shown how ontologies and background knowledge can aid each step of the KDD process. They perform association mining using the LISp-Miner tool, over the STULONG medical dataset. To support the data mining

they use UMLS ontologies[32] to map the data to semantic concepts. The mapping helped the authors to better understand the domain. They were able to identify and filter out redundant and unnecessary attributes, and group together semantically related attributes, by analyzing the relationships inside the ontology. Furthermore, they use ontologies to interpret and to give better explanation of the data mining results.

Table 3.4 gives an overview of the discussed approaches in this section. It shows that, while Linked Open Data based datasets play a minor role in this step, heavy-weight ontologies and reasoning are quite frequently used. Moreover, most of the ontologies are domain-independent, while domain-specific developments at this step are rather rare.

Ontologies are often used for supporting the user in creating a proper data mining process. They can be used to represent data sources, algorithms etc. in data mining processes, and assist the user in building reasonable data mining processes, e.g., by ensuring that a chosen algorithm is capable of handling the given data.
For example, the platform *RapidMiner* internally uses semantic descriptions of operators to assist the user in avoiding errors, e.g., when combining data preprocessing and machine learning operators. Here, reasoning does not only check the validity of a process, but also proposes solutions to fix an invalid process.
Approaches that use semantic information directly in an algorithm to influence the outcome of that algorithm are rather rare. There are some directions of using semantic background knowledge in data mining algorithms, e.g., for finding patterns that are easier to consume by an end user.

## 3.5 Interpretation

After the data mining step has been applied, we expect to discover some hidden patterns from the data. To be interpreted and understood, these patterns often require the use of some background knowledge, which is not always straightforward to find. In most real world contexts, providing the background knowledge is committed to the experts, whose work is to analyze the results of a data mining process, give them a meaning and refine them. The interpretation turns out to be an intensive and time-consuming process, where part of knowledge can remain unrevealed or unexplained.

Explain-a-LOD [219] is one of the first approaches in the literature for automatically generating hypothesis for explaining statistics by using LOD. The tool uses FeGeLOD (described in Section 3.3.1) for enhancing statistical datasets with background information from DBpedia, and uses correlation analysis and rule learining for producing hypothesis which are presented to the user. The tool has been later used to find and explain hypothesis for quality of living in cities across the world

[32] `http://www.nlm.nih.gov/research/umls/.`

Table 3.4: Summary of approaches used in the data mining step.

| Approach | Domain | | Ontology | | LOD | | Used Datasets | |
|---|---|---|---|---|---|---|---|---|
| | Problem | Data Mining | Complexity | Reasoning | Links | Semantics | LOD | Ontology |
| [14] | commerce | classification, regression, neural networks | L | no | no | no | / | custom ontology |
| [331, 332, 333, 238] | genomics, engineering, | classification | H | yes | no | no | / | KD |
| [68, 69, 70] | / | / | H | yes | no | no | / | KDDONTO |
| [146, 148, 147, 149] | socio-economy | clustering, | H | yes | no | no | / | DMO, DMWF,DMOP |
| [119, 120] | medicine | classification | H | yes | no | no | / | DMO, DMOP |
| [212, 213, 214, 215] | chemistry, pharmacology | classification, regression | H | yes | no | no | / | OntoDM, OntoDT, OntoDM-core, OntoDM-KDD |
| [94] | / | classification rule learning | L | no | no | no | / | WordNet |
| [235] | / | / | L | no | no | no | / | custom ontology |
| [210, 209] | / | / | L | no | no | no | / | custom ontology |

[220], and unemployment rates in France [251], among others. For example, in [220] the tool was able to automatically discover hypothesis like "Cities where many things take place have a high quality of living." and "European capitals of culture have a high quality of living.". While in [251] where data from DBpedia, Eurostat and LinkedGeoData has been used, the tool discovered hypothesis like "Regions in France that have high energy consumption have low unemployment rate." and "French regions that are out of Europe, French African Islands, and French Islands in the Indian Ocean have high unemployment rate.". Furthermore, the approach is extended in [250], which allows automatic correlation analysis and visualizing statistical data on maps using Linked Open Data. The tool allows the users to import any statistical data from local spreadsheets or RDF datacubes, perform correlation analysis and automatically visualize the findings on a map.

Linked Open Data cannot only add categorical information which allows for an easier exploration of results, but also additional visual clues. In [251, 250], we have shown that polygon data for geographic entities published as LOD, like GADM[33] can be exploited for creating a map-based visualization of data mining results. Moreover, GADM offers shape data of geographical entities on different administrative levels, which can be accessed through DBpedia by following *owl:sameAs* links.

d'Aquin et al. [51] have proposed a method that exploits external information available as LOD to support the interpretation of data mining results, through automatically building a navigation-exploration structure in the results of a particular type of data mining, in this case sequential pattern mining, tool based on data dimensions chosen by the analyst. To do so, the authors first represent the data mining results into a way compatible with a LOD representation, and link them to existing LOD sources. Then, the analyst can easily explore the mined results with additional dimension. Furthermore, to organize the enriched results into a hierarchy,the authors use formal concept analysis to build a concept lattice. This can allow the analyst to drill down into the details of a sub-set of the patterns, and see how it relates to the original data.

A similar approach is used in [132] for interpreting sequential patterns in patient data. Linked Data is used to support the interpretation of patterns mined from patient care trajectories. Linked data exposed through the BioPortal system is used to create a navigation structure within the patterns obtained form sequential pattern mining. The approach provides a flexible way to explore data about trajectories of diagnoses and treatments according to different medical classifications.

Tiddi [298] proposes a three step approach for interpreting data mining results, i.e. clustering, association rules and sequence patterns. In the first step additional information for the patterns results are extracted from the LOD cloud. Using inductive logic programming, new hypotheses are generated from the pattern mining results and the knowledge extracted from LOD. In the last step the hypotheses are evaluated using ranking strategies, like Weighted Relative Accuracy, and Informa-

[33] `http://gadm.geovocab.org/`

tion Retrieval F-measure. The same approach has been used in [299] to explain why groups of book, obtained from a clustering process, have been borrowed by the same students. The analysis were done on the Huddersfield's books usage dataset[34], using the British National Bibliography[35] and Library of Congress[36] as LOD datasets. The experiments lead to interesting hypothesis to explain the clusters, e.g., "books borrowed by students of Music Technologies are clustered together because they talk about music".

The work has been continued in [300, 302], introducing *Dedalo*, framework that dynamically traverses Linked Data to find commonalities that form explanations for items of a cluster. Dedalo uses iterative approach for traversing LOD graphs, where the roots are the items of the clusters. The underlying assumption is that items that belong to one cluster share more common paths in the LOD graph, than the items outside the cluster. The authors were able to extract interesting and representative explanation for the clusters, however, the number of resulting atomic rules is rather large, and need to be aggregated in a post-processing step. The typical strategy to overcome those problems is providing the patterns to human experts, whose role consists in analysing the results, discovering the interesting ones while explaining, pruning or refining the unclear ones. To cope with such a strenuous and time consuming process, the authors in their next work [301] have proposed an approach that is using Neural Network model to predict whether two rules, if combined, can lead to the creation of a new, improved rule (i.e., a new rule, with a better F-measure). The approach has been applied in the domain of education and publications.

Lavrač et al. have made a notable research work in the field of semantic subgroup discovery. The task of subgroup discovery is defined as follows: "Given a population of individuals and a property of those individuals that we are interested in, find population subgroups that are statistically most interesting, for example, are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest" [151]. The authors define semantic subgroup discovery as part of semantic data mining, which is defined as: "Given a set of domain ontologies, and empirical data annotated by domain ontology terms, one can find a hypothesis (a predictive model or a set of descriptive patterns), expressed by domain ontology terms, explaining the given empirical data". The semantic subgroup discovery was first implemented in the SEGS system [305]. SEGS uses as background knowledge data from three publicly available, semantically annotated biological data repositories. Based on the background knowledge, it automatically formulates biological hypotheses: rules which define groups of differentially expressed genes. Finally, it estimates the relevance (or significance) of the automatically formulated hypotheses on experimental microarray data. The system was extended in the SegMine system, which allows exploratory

[34] http://library.hud.ac.uk/data/usagedata/readme.html
[35] http://bnb.data.bl.uk/
[36] http://id.loc.gov/

analysis of microarray data, performed through semantic subgroup discovery by SEGS [237], followed by link discovery and visualization by Biomine [78], an integrated annotated bioinformatics information resource of interlinked data. The SEGS system was later extended to two general semantic subgroup discovery systems, SDM-SEGS and SDM-Aleph [201, 159, 313]. Finally, the authors introduced the Hedwig system [314], which overcomes some of the limitations of the previous systems. The findings of this series of work has been concluded in [158, 315].

Many approaches are using ontologies for patterns post-mining, and interpretation of the results. The domain knowledge and metadata specification stored in the ontology are used in the interpretation stage to prune and filter out discovered patterns. Ontologies are commonly used to filter out redundant patterns, and too specific patterns without losing semantic information. One of the first approaches that uses domain ontologies for that purpose is the work by Srikant [286], who introduced the concept of generalized association rules. Similarly, Zhou et al. [340] introduce the concept of raising. Raising is the operation of generalizing data mining rules in order to increase the support while keeping the confidence high enough. This is done with generalizing the entities by raising them to a specified level in the ontology. The authors use an ontology that consist of two taxonomies, one of which describes different customer classifications, while the other one contains a large hierarchy, based on Yahoo, which contains interests. In the experiments, the support values of rule sets were greatly increased, up to 40 times. GART is a very similar approach [72], which uses several taxonomies over attributes to iteratively generalize rules, and then, prune redundant rules at each step. The experiments were performed using a sale database of a Brazilian supermarket. The experiments show reduction rates of the sets of association rules varying from 14,61% to 50,11%. Marninica et al. [173] presents an interactive postprocessing framework, called ARIPSO (Association Rule Interactive post-Processing using Schemas and Ontologies). The framework assists the user throughout the analyzing task to prune and filter discovered rules. The system allows formalizing user knowledge and goals, which are latter used in applying iteratively a set of filters over extracted rules in order to extract interesting rules: minimum improvement constraint filter, item-relatedness filter, rule schema filters/pruning. The experiments were performed on the Nantes Habitat data[37], dealing with customers satisfaction concerning accommodation, for which a corresponding ontology was developed by the authors. The results showed that the number of rules can be significantly reduced when using the schema, resulting in more descriptive rules.

Huang et al. [129] use LOD to interpret the results of text mining. The approach starts with extracting entities and semantic relations between them from text documents, resulting into semantic graphs. Then, a frequent sub-graph discovery algorithm is applied on the text graphs to find frequent patterns. To interpret the discovered subgraphs, an algorithm is proposed to traverse Linked Data graphs for relations that are used to annotate the vertices and the edges of the frequent sub-

[37] http://www.nantes-habitat.fr/

graphs. The approach is applied on a military dataset, where DBpedia is used as a LOD dataset.

Another approach that uses ontologies in rule mining is the 4ft-Miner tool [293]. The tool is used in four stages of the KDD process: data understanding, data mining, result interpretation and result dissemination. In the data understanding step, a data-to-ontology mapping was performed, which resulted in discovery of redundant attributes. In the data mining stage of the KDD process, the ontology was used to decompose the data mining task into more specific tasks, which can be run faster, resulting in more homogeneous results, and thus easily interpretable. In the interpretation stage, the data-to-ontology mappings are used to match some of the discovered associations to the corresponding semantic relations or their more complex chains from the ontology, which can be considered as potential explanation of the discovered associations. The approach was used to interpret associations in medical and social climate applications. In the medical domain, the STULONG dataset[38] is used, which contains cardiovascular risk data. As an ontology is used the UMLS ontology. Using the approach, the authors were able to discover hypothesis like "Patients who are not physically active within the job nor after the job (Antecedent) will more often have higher blood pressure (Succedent)" and ""Increase of smoking leads to increase of cardio-vascular diseases".

Table 3.5 gives an overview of the discussed approaches in this section. We observe that in this step, reasoning plays no crucial role. The datasets exploited are rather mixed, general purpose datasets such as DBpedia are often used, but also highly specific datasets can be exploited. Roughly half of the approaches also make use of the interlinks between those datasets.

Semantic Web data can help in the interpretation of patterns found, in particular for descriptive tasks. Those typically encompass subgroups or clusters found, or rule models that are used to describe a dataset.
The information used from LOD datasets and/or ontologies can help further analyzing those findings, e.g., by explicating typical features of instances in a subgroup or cluster, thus, they may explain the grouping chosen by a data mining algorithm. Furthermore, rules can be further refined and/or generalized, which improves their interpretability.

## 3.6 Discussion

Given the amount of research works discussed in this chapter, it is evident that, especially with the advent and growth of Linked Open Data, information from the Semantic Web can be used beneficially in the data mining and knowledge discovery process. Looking at the results from a larger distance, however, we can make a few interesting observations:

- DBpedia is used in the vast majority of the research papers discussed in this survey, with other LOD sources being used only scarcely, and the majority of

[38] http://euromise.vse.cz/stulong-en/

Table 3.5: Summary of approaches used in the interpretation step.

| Approach | Domain | | Ontology | | LOD | | Used Datasets | |
|---|---|---|---|---|---|---|---|---|
| | Problem | Data Mining | Complexity | Reasoning | Links | Semantics | LOD | Ontology |
| [219] | sociology, economy | pattern mining | H | no | yes | yes | DBpedia | / |
| [220, 251, 250] | statistics | pattern mining | H | no | yes | yes | DBpedia, Eurostat, Linked-GeoData, GADM | / |
| [51, 132] | students, medicine | pattern mining | H | no | yes | yes | Open University's course catalog[a], ICD10, CCAM Bio Ontology[b] | / |
| [298, 299] | books, publications | clustering, association rules | H | no | yes | no | British National Bibliography[c], Library of Congress[d] | / |
| [300, 302, 301] | education, publication | clustering | H | no | yes | no | DBpedia, UIS[e], British National Bibliography, Library of Congress | / |
| [305, 237, 78, 201, 159, 313, 315] | biomedicine | rule learning, subgroup discovery | H | no | no | no | / | GO[f], KEGG[g], Entrez |
| [265] | / | outlier detection | H | no | no | no | DBpedia | / |
| [314] | finance | subgroup discovery | H | no | yes | no | GeoNames | / |
| [340] | commerce | rule learning | H | no | no | no | / | interest ontology(from Yahoo) |
| [72] | commerce | rule learning | H | no | no | no | / | products taxonomy |
| [173] | sociology | rule learning | H | no | no | no | / | custom ontology |
| [129] | military | subgroup discovery | H | no | yes | yes | DBpedia | / |
| [293] | medicine, sociology | association mining | H | yes | no | no | no | UMLS, Social climate ontology |

the hundreds of LOD datasets not being used at all. There may be different reasons for that; ranging from DBpedia's relatively simple data model and its wide coverage to the availability of sophisticated tools such as DBpedia Lookup and DBpedia Spotlight.
While this underlines the utility of such general purpose knowledge sources on the Semantic Web, it can also be problematic to tailor and evaluate approaches only to single datasets, since it limits the insights on the general applicability of the approaches.

- Many papers use custom ontologies and datasets instead of reusing open datasets from the web of data. This is particularly often observed in the life sciences and medical domain, which, at the same time, is one of the largest most prominently represented domains within the Linked Open Data cloud. It is subject to future research to find out the reasons for this discrepancy, which may have different reasons, such as a limited awareness of open datasets, or an inferior fitness for use of those datasets.

- Links between datasets, which are one of the core ingredients to *Linked* Open Data, are used by relatively few approaches. This may also imply that many of the approaches stay below what is possible with Linked Open Data, leveraging only information from one dataset instead of using the full amount of knowledge captured in the Semantic Web. One reason may be that even in the presence of machine-interpretable schemas, developing schema-agnostic applications is a non-trivial task. Furthermore, building approaches that autonomously follow links and are ultimately capable of exploiting the whole Web of Linked Data as background knowledge would also lead to new scalability challenges.

- Expressive schemas/ontologies and reasoning on those, which has been a core selling point of the Semantic Web for years, are rarely combined with data mining and knowledge discovery. Again, it is subject to future research to find out whether this is due to a limited availability of suitable ontologies, limited awareness, or imperfect fitness to the problems found in practice.

- In most cases, knowledge from the Semantic Web is about the domain of the processed data, not the data mining domain. However, given endpoints such as *myExperiment.org*[39], which provides lots of scientific workflows (including data mining workflows), would allow for solutions providing advice to data analysts building such workflows, such as the recently announced "Wisdom of Crowds Operator Recommendations" by RapidMiner[40], based on open data.

These observations show that, although a remarkable amount of work in the area exists, data mining and knowledge discovery is still not tapping the full potential

[39] `http://www.myexperiment.org`

[40] `https://rapidminer.com/news-posts/rapidminer-makes-snap-move-predictive-analytics-data-mining-machine-learning-cloud/`

that is provided by the Semantic Web. Data mining workflows automatically leveraging information from different datasets by following links beyond single datasets such as DBpedia are still an interesting and promising area of research.

## 3.7 Conclusion and Outlook

In this chapter, we have given an overview of the usage of Semantic Web data, most prominently Linked Open Data, for data mining and knowledge discovery. Following Fayyad's classic workflow pipeline, we have shown examples for the usage of Semantic Web data at every stage of the pipeline, as well as approaches supporting the full pipeline.

Analyzing the findings, the first observation is that there are plenty of works of research in the area, and applications exist in many domains. A frequent application domain is biomedicine and life science, but the approaches are also transferred to quite a few other domains. Furthermore, some sophisticated applications and tool stacks exist, that go beyond mere research prototypes.

Furthermore, we see that there are still some uncharted territories in the research landscape of Semantic Web enabled data mining. This shows that, although impressive results can be achieved already today, the full potential of Semantic Web enabled data mining and KDD still remains to be unlocked.

# Part I

# Mining Semantic Web Knowledge Graphs

# Chapter 4

# A Collection of Benchmark Datasets for Systematic Evaluations of Machine Learning on the Semantic Web

In the previous chapter, we showed that in the recent years, applying machine learning to Semantic Web data has drawn a lot of attention. Many approaches have been proposed for different tasks at hand, ranging from reformulating machine learning problems on the Semantic Web as traditional, propositional machine learning tasks to developing entirely novel algorithms. However, systematic comparative evaluations of different approaches are scarce; approaches are rather evaluated on a handful of often project-specific datasets, and compared to a baseline and/or one or two other systems.

In contrast, evaluations in the machine learning area are often more rigorous. Approaches are usually compared using a larger number of standard datasets, most often from the UCI repository[1]. With a larger set of datasets used in the evaluation, statements about statistical significance are possible as well [60].

At the same time, collections of benchmark datasets have become quite well accepted in other areas of Semantic Web research. Notable examples include the Ontology Alignment Evaluation Initiative (OAEI) for ontology matching[2], the *Berlin SPARQL Benchmark*[3] for triple store performance, the Lehigh University Benchmark (LUBM)[4] for reasoning, or the Question Answering over Linked Data (QALD) dataset[5] for natural language query systems.

[1] http://archive.ics.uci.edu/ml/

[2] http://oaei.ontologymatching.org/

[3] http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/

[4] http://swat.cse.lehigh.edu/projects/lubm/

[5] http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/

Furthermore, there are some challenges, like the *Linked Data Mining Challenge* [256] or the *Semantic-Web enabled Recommender Systems Challenge* [63], which usually focus on only a few datasets and a very specific problem setting. A quantitative comparison of machine learning approaches on a larger set of datasets that allow for statistically significant conclusions, has not been proposed so far. To the best of our knowledge, there is no publicly accessible collection of datasets that allow for such a comparison.

In this chapter, we introduce a collection of datasets for benchmarking machine learning approaches for the Semantic Web. Those datasets are either existing RDF datasets, or external classification or regression problems, for which the instances have been enriched with links to the Linked Open Data cloud [267]. Furthermore, by varying the number of instances for a dataset, scalability evaluations are also made possible.

**The work presented in this chapter has been published before as: "Petar Ristoski, Gerben Klaas Dirk de Vries, Heiko Paulheim: *A Collection of Benchmark Datasets for Systematic Evaluations of Machine Learning on the Semantic Web.* Proceedings of the 15th International Semantic Web Conference, Kobe, Japan, October, 2016." [247].**

## 4.1 Datasets

Our dataset collection has three categories: (i) existing datasets that are commonly used in machine learning experiments, (ii) datasets that were generated from official observations, and (iii) datasets generated from existing RDF datasets. Each of the datasets in the first two categories are initially linked to DBpedia[6]. This has two main reasons, (1) DBpedia being a cross-domain knowledge base usable in datasets from very different topical domains, and (2) tools like DBpedia Lookup and DBpedia Spotlight making it easy to link external datasets to DBpedia. However, DBpedia can be seen as an entry point to the Web of Linked Data, with many datasets linking to and from DBpedia. In fact, we use the RapidMiner Linked Open Data extension [246], to retrieve external links for each entity to YAGO[7] and Wikidata[8]. Such links could be exploited for systematic evaluation of the relevance of the data of different LOD dataset in different learning tasks.

In the dataset collection, there are four datasets that are commonly used for machine learning. For these datasets, we first enrich the instances with links to LOD datasets, and reuse the already defined target variable to perform machine learning experiments:

[6] http://dbpedia.org
[7] http://yago-knowledge.org/
[8] http://www.wikidata.org

- The *Auto MPG* dataset[9] captures different characteristics of cars, and the target is to predict the fuel consumption (MPG) as a regression task.

- The *AAUP* (American Association of University Professors) dataset contains a list of universities, including eight target variables describing the salary of different staff at the universities[10]. We use the average salary as a target variable both for regression and classification, discretizing the target variable into "high", "medium" and "low", using equal frequency binning.

- The *Auto 93* dataset[11] captures different characteristics of cars, and the target is to predict the price of the vehicles as a regression task.

- The *Zoo* dataset captures different characteristics of animals, and the target is to predict the type of the animals as a classification task.

For those datasets, cars, universities, and animals are linked to DBpedia based on their name.

The second category of datasets contains a list of datasets where the target variable is an observation from different real-world domains, as captured by official sources. Again, the instances were enriched with links to LOD datasets. There are thirteen datasets in this category:

- The *Forbes* dataset contains a list of companies including several features of the companies, which was generated from the Forbes list of leading companies 2015[12]. The target is to predict the company's market value as a classification and regression task. To use it for the task of classification we discretize the target variable into "high", "medium", and "low", using equal frequency binning.

- The *Cities* dataset contains a list of cities and their quality of living, as captured by Mercer [219]. We use the dataset both for regression and classification.

- The *Endangered Species* dataset classifies animals into endangered species[13].

- The *Facebook Movies* dataset contains a list of movies and the number of Facebook likes for each movie[14]. We first selected $10,000$ movies from DBpedia, which were then linked to the corresponding Facebook page, based on the movie's name and the director. The final dataset contains $1,600$ movies, which was created by first ordering the list of movies based on the number of Facebook likes, and then selecting the top $800$ movies and the bottom $800$ movies. We use the dataset for regression and classification.

[9] `http://archive.ics.uci.edu/ml/datasets/Auto+MPG`

[10] `http://www.amstat.org/publications/jse/jse_data_archive.htm`

[11] `http://www.amstat.org/publications/jse/v1n1/datasets.lock.html`

[12] `http://www.forbes.com/global2000/list/`

[13] `http://a-z-animals.com/`

[14] We use the Facebook Graph API: `https://developers.facebook.com/docs/graph-api`

- Similarly, the *Facebook Books* dataset contains a list of books and the number of Facebook likes. Each book was linked to the corresponding Facebook page using the book's title and the book's author. Again, we selected the top $800$ books and the bottom $800$ books, based on the number of Facebook likes.
- The *Metacritic Movies* dataset is retrieved from Metacritic.com[15], which contains an average rating of all time reviews for a list of movies [256]. The initial dataset contained around $10,000$ movies, from which we selected $1,000$ movies from the top of the list, and $1,000$ movies from the bottom of the list. We use the dataset both for regression and classification.
- Similarly, the *Metacritic Albums* dataset is retrieved from Metacritic.com[16], which contains an average rating of all time reviews for a list of albums [257].
- The *HIV Deaths Country* dataset contains a list of countries with the number of deaths caused by HIV, as captured by the World Health Organization[17]. We use the dataset both for regression and classification.
- Similarly, the *Traffic Accidents Deaths Country* dataset contains a list of countries with the number of deaths caused by traffic accidents[18].
- The *Energy Savings Country* dataset contains a list of countries with the total amount of energy savings of primary energy in 2010[19], which was downloaded from WorldBank[20]. We use the dataset both for regression and classification.
- Similarly, the *Inflation Country* dataset contains a list of countries with the inflation rate for 2011[21].
- The *Scientific Journals Country* dataset contains a list of countries with a number of scientific and technical journal articles published in 2011[22].
- The *Unemployment French Region* dataset contains a list of regions in France with the unemployment rate, used in the SemStats 2013 challenge [251].

Again, for those datasets, the instances (cities, countries, etc.) are linked to DBpedia. For datasets which are used for classification and regression, the regression target was discretized using equal frequency binning, usually into a *high* and a *low* class.

The third, and final, category contains datasets that were generated from existing RDF datasets, where the value of a certain property is used as a classification target. There are five datasets in this category:

[15] `http://www.metacritic.com/browse/movies/score/metascore/all`
[16] `http://www.metacritic.com/browse/albums/score/metascore/all`
[17] `http://apps.who.int/gho/data/view.main.HIV1510`
[18] `http://apps.who.int/gho/data/view.main.51310`
[19] `http://data.worldbank.org/indicator/10.1_ENERGY.SAVINGS`
[20] `http://www.worldbank.org/`
[21] `http://data.worldbank.org/indicator/NY.GDP.DEFL.KD.ZG`
[22] `http://data.worldbank.org/indicator/IP.JRN.ARTC.SC`

- The *Drug-Food Interaction* dataset contains a list of drug-recipe pairs and their interaction, i.e., "negative" and "neutral" [135]. The dataset was retrieved from FinkiLOD[23]. Furthermore, each drug is linked to DrugBank[24]. We drew a stratified random sample of $2,000$ instances from the complete dataset. When generating the features, we ignore the `foodInteraction` property in DrugBank, since it highly correlates with the target variable.

- The *AIFB* dataset describes the AIFB research institute in terms of its staff, research group, and publications. In [23] the dataset was first used to predict the affiliation (i.e., research group) for people in the dataset. The dataset contains 178 members of a research group, however the smallest group contains only 4 people, which is removed from the dataset, leaving 4 classes. Also, we remove the `employs` relation, which is the inverse of the *affiliation* relation.

- The *AM* dataset contains information about artifacts in the Amsterdam Museum [53]. Each artifact in the dataset is linked to other artifacts and details about its production, material, and content. It also has an artifact category, which serves as a prediction target. We have drawn a stratified random sample of $1,000$ instances from the complete dataset. We also removed the `material` relation, since it highly correlates with the artifact category.

- The *MUTAG* dataset is distributed as an example dataset for the DL-Learner toolkit[25]. It contains information about complex molecules that are potentially carcinogenic, which is given by the `isMutagenic` property.

- The *BGS* dataset was created by the British Geological Survey and describes geological measurements in Great Britain[26]. It was used in [57] to predict the lithogenesis property of named rock units. The dataset contains 146 named rock units with a lithogenesis, from which we use the two largest classes.

An overview of the datasets is given in Tables 4.1, 4.2, and 4.3. For each dataset, we depict the number of instances, the machine learning tasks in which the dataset is used (*C* stands for classification and *R* stands for regression), the source of the dataset, and the LOD datasets to which the dataset is linked. For each dataset, we depict basic statistics of the properties of the LOD datasets, i.e., average, median, maximum and minimum number of *types*, *categories*, *outgoing relations* (rel out), *incoming relations* (rel in), outgoing relations including values (rel-vals out) and incoming relations including values (rel-vals in). The datasets, as well as a detailed description, a link quality evaluation, and licensing information, can be found online[27].

[23] `http://linkeddata.finki.ukim.mk/`
[24] `http://wifo5-03.informatik.uni-mannheim.de/drugbank/`
[25] `http://dl-learner.org`
[26] `http://data.bgs.ac.uk/`
[27] `http://w3id.org/sw4ml-datasets`

From the given statistics, we can infer the following observations: (i) DBpedia contains significantly less *owl:sameAs* links to YAGO, compared to Wikidata; (ii) DBpedia provides the highest number of types and categories on average per entity; (iii) Wikidata contains the highest number of outgoing and incoming relations for most of the datasets; (iv) YAGO contains the highest number of outgoing and incoming relations values for most of the datasets.

## 4.2 Experiments

To demonstrate how the dataset collection can be used for systematic evaluations, we set up an experiment as follows: we compare different propositional machine learning approaches both for classification and regression, each combined with various strategies for converting the LOD instances to a propositional feature vector. With this setup, we demonstrate how a systematic comparison between different propositionalization strategies can be carried out.

It is important to note that this section is *not* meant to be an ultimate comparison of machine learning techniques for the Semantic Web, but rather as an example on how to create structured evaluations with the benchmark collection at hand. However, we will be able to draw some conclusions about the performance of different feature generation strategies and classifiers at the end of the section.

### 4.2.1 Feature Generation Strategies

For generating the data mining features, we use three strategies that take into account the direct relations to other resources in the graph [225, 252], and two strategies for features derived from graph sub-structures [58, 59]:

- Features derived from specific relations. In the experiments we use the relations *rdf:type* (types), and *dcterms:subject* (categories) for datasets linked to DBpedia.
- Features derived from generic relations, i.e., we generate a feature for each incoming (rel in) or outgoing relation (rel out) of an entity, ignoring the value of the relation.
- Features derived from generic relations-values, i.e, we generate feature for each incoming (rel-vals in) or outgoing relation (rel-vals out) of an entity including the value of the relation.
- Kernels that count substructures in the RDF graph around the instance node. These substructures are explicitly generated and represented as sparse feature vectors.
  - The Weisfeiler-Lehman (WL) graph kernel for RDF [57] counts full subtrees in the subgraph around the instance node. This kernel has two parameters, the subgraph depth $d$ and the number of iterations $h$ (which determines the depth of the subtrees). We use two pairs of settings, $d = 1, h = 2$ and $d = 2, h = 3$.

Table 4.1: Datasets statistics I

| Dataset | | | | types | | | | categories | | | | rel out | | | | rel in | | | | rel-vals out | | | | rel-vals in | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Source | Task | LOD | #links | avg | med | max | min | avg | med | max | min | avg | med | max | min | avg | med | max | min | avg | med | max | min | avg | med | max | min |
| Auto MPG | UCI ML | R | DBpedia | 371 | 29.70 | 31 | 46 | 5 | 11.20 | 10 | 25 | 2 | 13.48 | 13 | 27 | 3 | 5.62 | 5 | 25 | 1 | 16.50 | 15 | 70 | 0 | 36.65 | 23 | 509 | 0 |
| | | | YAGO | 331 | 13.99 | 16 | 21 | 0 | 9.26 | 9 | 23 | 0 | 8.76 | 9 | 18 | 0 | 16.96 | 2 | 138 | 1 | 77.08 | 70 | 278 | 0 | 3,236.24 | 60 | 28,418 | 0 |
| | | | Wikidata | 371 | 1.05 | 1 | 3 | 0 | 0.29 | 0 | 3 | 0 | 20.20 | 18 | 61 | 9 | 5.32 | 5 | 31 | 1 | 13.92 | 12 | 54 | 4 | 59.33 | 21 | 755 | 3 |
| AAUP | JSE | R/C (c=3) | DBpedia | 960 | 24.40 | 28 | 41 | 0 | 9.38 | 9 | 20 | 0 | 12.68 | 15 | 28 | 0 | 8.20 | 7 | 36 | 0 | 11.74 | 11 | 66 | 0 | 62.18 | 23 | 2,488 | 0 |
| | | | YAGO | 889 | 10.49 | 11 | 17 | 0 | 3.31 | 3 | 11 | 0 | 11.37 | 12 | 15 | 0 | 13.61 | 3 | 138 | 1 | 85.83 | 68 | 446 | 0 | 2,455.27 | 110 | 28,418 | 1 |
| | | | Wikidata | 959 | 2.13 | 2 | 5 | 0 | 0.88 | 1 | 2 | 0 | 30.71 | 29 | 83 | 0 | 8.51 | 7 | 44 | 0 | 22.38 | 21 | 97 | 0 | 296.92 | 20 | 31,777 | 0 |
| Auto 93 | JSE | R | DBpedia | 93 | 28.76 | 31 | 43 | 5 | 11.13 | 10 | 25 | 3 | 12.69 | 12 | 22 | 8 | 4.92 | 5 | 7 | 2 | 14.35 | 11 | 64 | 4 | 22.60 | 18 | 64 | 2 |
| | | | YAGO | 80 | 13.80 | 16 | 19 | 0 | 9.09 | 10 | 18 | 0 | 8.37 | 10 | 11 | 0 | 21.09 | 2 | 138 | 2 | 59.33 | 59 | 129 | 0 | 4,025.90 | 46 | 28,418 | 4 |
| | | | Wikidata | 93 | 1.00 | 1 | 2 | 0 | 0.12 | 0 | 1 | 0 | 17.31 | 17 | 26 | 9 | 3.56 | 3 | 8 | 1 | 11.23 | 11 | 25 | 4 | 19.91 | 19 | 57 | 3 |
| Zoo | UCI ML | C (c=3) | DBpedia | 101 | 8.61 | 11 | 26 | 0 | 4.67 | 3 | 34 | 0 | 8.22 | 9 | 15 | 3 | 3.54 | 3 | 8 | 1 | 13.26 | 11 | 87 | 1 | 146.28 | 24 | 3,686 | 2 |
| | | | YAGO | 8 | 0.74 | 0 | 13 | 0 | 0.15 | 0 | 6 | 0 | 0.63 | 1 | 8 | 0 | 127.23 | 138 | 138 | 2 | 5.39 | 1 | 156 | 0 | 26,173.23 | 28,418 | 28,418 | 3 |
| | | | Wikidata | 101 | 1.00 | 1 | 2 | 0 | 0.67 | 1 | 2 | 0 | 29.69 | 35 | 57 | 3 | 8.28 | 7 | 27 | 0 | 18.20 | 21 | 45 | 1 | 125.82 | 92 | 785 | 0 |
| Forbes | Forbes | R/C (c=2) | DBpedia | 1,585 | 14.77 | 19 | 62 | 0 | 4.87 | 4 | 52 | 0 | 10.15 | 11 | 27 | 0 | 2.76 | 2 | 27 | 0 | 10.44 | 10 | 136 | 0 | 14.30 | 4 | 1,925 | 0 |
| | | | YAGO | 1,003 | 7.28 | 10 | 33 | 0 | 2.35 | 2 | 42 | 0 | 7.57 | 11 | 21 | 0 | 52.07 | 2 | 138 | 1 | 34.42 | 27 | 510 | 0 | 10,531.37 | 107 | 28,418 | 1 |
| | | | Wikidata | 1,189 | 0.82 | 1 | 4 | 0 | 0.22 | 0 | 3 | 0 | 16.59 | 16 | 137 | 0 | 5.00 | 5 | 52 | 0 | 12.69 | 10 | 207 | 0 | 30.14 | 8 | 2,881 | 0 |
| Cities | Mercer | R/C (c=3) | DBpedia | 212 | 31.28 | 35 | 53 | 0 | 6.98 | 7 | 26 | 0 | 18.08 | 19 | 38 | 0 | 25.66 | 25 | 68 | 0 | 16.26 | 13 | 131 | 0 | 1,474.57 | 678 | 19,810 | 0 |
| | | | YAGO | 187 | 16.66 | 19 | 30 | 0 | 4.46 | 4 | 15 | 0 | 13.75 | 15 | 32 | 0 | 23.56 | 9 | 138 | 2 | 222.54 | 214 | 681 | 0 | 8,087.34 | 3,555 | 72,320 | 5 |
| | | | Wikidata | 212 | 2.11 | 2 | 9 | 1 | 3.40 | 4 | 6 | 0 | 69.08 | 67 | 153 | 6 | 39.99 | 37 | 108 | 1 | 105.29 | 89 | 390 | 2 | 5,298.23 | 1,599 | 99,865 | 1 |
| FB Books | Facebook | R/C (c=2) | DBpedia | 1,600 | 19.08 | 20 | 42 | 0 | 5.15 | 5 | 23 | 0 | 11.15 | 11 | 20 | 0 | 1.64 | 2 | 7 | 0 | 7.04 | 7 | 60 | 0 | 2.80 | 2 | 42 | 0 |
| | | | YAGO | 1,334 | 8.37 | 10 | 24 | 0 | 2.03 | 2 | 15 | 0 | 8.41 | 10 | 13 | 0 | 25.32 | 3 | 138 | 1 | 24.37 | 22 | 149 | 0 | 4,735.50 | 8 | 28,418 | 1 |
| | | | Wikidata | 1,578 | 1.00 | 1 | 3 | 0 | 0.01 | 0 | 1 | 0 | 21.19 | 22 | 55 | 0 | 3.15 | 3 | 17 | 0 | 16.41 | 16 | 69 | 0 | 7.47 | 4 | 165 | 0 |
| FB Movies | Facebook | R/C (c=2) | DBpedia | 1,600 | 24.90 | 27 | 55 | 0 | 12.50 | 11 | 60 | 0 | 12.43 | 13 | 21 | 0 | 1.46 | 1 | 12 | 0 | 11.65 | 12 | 51 | 0 | 4.96 | 2 | 110 | 0 |
| | | | YAGO | 1,339 | 12.08 | 14 | 32 | 0 | 6.51 | 6 | 27 | 0 | 8.39 | 10 | 17 | 0 | 26.89 | 6 | 138 | 1 | 55.01 | 47 | 280 | 0 | 4,682.42 | 43 | 28,418 | 1 |
| | | | Wikidata | 1,585 | 1.01 | 1 | 4 | 0 | 0.04 | 0 | 1 | 0 | 48.75 | 48 | 107 | 0 | 2.22 | 1 | 22 | 0 | 56.37 | 53 | 372 | 0 | 20.75 | 12 | 230 | 0 |
| Metacritic Albums | Metacritic | R/C (c=2) | DBpedia | 1,600 | 17.92 | 19 | 36 | 0 | 4.27 | 4 | 26 | 0 | 10.85 | 12 | 17 | 2 | 2.63 | 3 | 7 | 0 | 8.92 | 9 | 63 | 0 | 5.28 | 3 | 50 | 0 |
| | | | YAGO | 1,444 | 7.22 | 8 | 19 | 0 | 3.22 | 3 | 20 | 0 | 8.05 | 9 | 10 | 0 | 16.02 | 3 | 138 | 1 | 40.27 | 32 | 361 | 0 | 2,749.90 | 10 | 28,418 | 1 |
| | | | Wikidata | 1,576 | 0.99 | 1 | 2 | 0 | 0.00 | 0 | 1 | 0 | 17.64 | 18 | 45 | 0 | 4.00 | 5 | 9 | 1 | 11.73 | 12 | 49 | 0 | 8.77 | 7 | 54 | 1 |

Table 4.2: Datasets statistics II

| Dataset | | | | | types | | | | categories | | | | rel out | | | | rel in | | | | rel-vals out | | | | rel-vals in | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Source | Task | LOD | #links | avg | med | max | min | avg | med | max | min | avg | med | max | min | avg | med | max | min | avg | med | max | min | avg | med | max | min |
| Metacritic Movies | Metacritic | R/C (c=2) | DBpedia | 2,000 | 24.38 | 27 | 45 | 0 | 11.87 | 11 | 42 | 0 | 12.54 | 14 | 19 | 3 | 1.35 | 1 | 7 | 0 | 11.42 | 12 | 30 | 0 | 3.56 | 2 | 31 | 0 |
| | | | YAGO | 1,588 | 11.79 | 14 | 19 | 0 | 6.43 | 6 | 28 | 0 | 8.34 | 10 | 11 | 0 | 28.22 | 6 | 138 | 1 | 48.84 | 43 | 216 | 0 | 4,960.84 | 37 | 28,418 | 1 |
| | | | Wikidata | 1,981 | 0.98 | 1 | 1 | 0 | 0.03 | 0 | 1 | 0 | 47.86 | 49 | 99 | 0 | 1.98 | 1 | 13 | 0 | 52.70 | 53 | 237 | 0 | 15.77 | 11 | 117 | 0 |
| HIV Deaths Country | WHO | R/C (c=2) | DBpedia | 114 | 35.69 | 37 | 52 | 0 | 12.61 | 13 | 23 | 0 | 23.59 | 24 | 28 | 3 | 34.26 | 31 | 89 | 6 | 27.75 | 25 | 162 | 10 | 4,828.36 | 1,065 | 70,426 | 24 |
| | | | YAGO | 108 | 13.90 | 15 | 24 | 0 | 9.28 | 9 | 18 | 0 | 28.41 | 31 | 35 | 0 | 15.18 | 9 | 138 | 5 | 302.34 | 244 | 1,267 | 0 | 12,464.42 | 4,879 | 112,032 | 550 |
| | | | Wikidata | 114 | 4.12 | 4 | 8 | 1 | 4.83 | 5 | 6 | 0 | 120.87 | 119 | 173 | 7 | 55.68 | 51 | 148 | 2 | 229.46 | 210 | 595 | 2 | 45,671.15 | 4,971 | 669,273 | 66 |
| Trafic Accidents Country | WHO | R/C (c=2) | DBpedia | 146 | 36.40 | 38 | 53 | 0 | 13.12 | 13 | 23 | 0 | 23.40 | 24 | 28 | 1 | 37.87 | 36 | 94 | 8 | 27.44 | 24 | 162 | 0 | 7,528.18 | 1,587 | 218,957 | 77 |
| | | | YAGO | 139 | 14.29 | 15 | 27 | 0 | 9.62 | 10 | 16 | 0 | 28.44 | 31 | 35 | 0 | 14.61 | 9 | 138 | 5 | 345.03 | 290 | 2,104 | 0 | 17,882.47 | 6,126 | 423,559 | 693 |
| | | | Wikidata | 146 | 4.42 | 4 | 10 | 1 | 4.94 | 5 | 6 | 0 | 124.31 | 121 | 191 | 7 | 61.68 | 55 | 148 | 2 | 242.38 | 213 | 713 | 2 | 85,575.10 | 7,369 | 1,557,157 | 66 |
| Energy Savingss Country | WorldBank | R/C (c=2) | DBpedia | 162 | 36.07 | 38 | 53 | 0 | 13.12 | 13 | 23 | 0 | 23.46 | 24 | 28 | 1 | 36.64 | 33 | 94 | 8 | 26.72 | 23 | 162 | 0 | 6,876.80 | 1,440 | 218,957 | 77 |
| | | | YAGO | 152 | 14.09 | 15 | 27 | 0 | 9.52 | 10 | 16 | 0 | 27.82 | 31 | 35 | 0 | 16.40 | 9 | 138 | 5 | 329.28 | 279 | 2,104 | 0 | 16,969.96 | 5,821 | 423,559 | 757 |
| | | | Wikidata | 162 | 4.41 | 4 | 10 | 1 | 4.92 | 5 | 6 | 0 | 123.36 | 119 | 191 | 7 | 60.02 | 55 | 148 | 2 | 238.69 | 210 | 713 | 2 | 77,485.01 | 5,810 | 1,557,157 | 66 |
| Inflation Country | WorldBank | R/C (c=2) | DBpedia | 160 | 36.00 | 38 | 53 | 0 | 13.11 | 13 | 23 | 0 | 23.46 | 24 | 28 | 1 | 36.74 | 33 | 94 | 8 | 26.85 | 24 | 162 | 0 | 6,947.59 | 1,440 | 218,957 | 77 |
| | | | YAGO | 150 | 14.09 | 15 | 27 | 0 | 9.44 | 10 | 16 | 0 | 27.80 | 31 | 35 | 0 | 16.48 | 9 | 138 | 5 | 331.16 | 279 | 2,104 | 0 | 17,114.88 | 5,821 | 423,559 | 693 |
| | | | Wikidata | 160 | 4.39 | 4 | 10 | 1 | 4.88 | 5 | 6 | 0 | 123.23 | 119 | 191 | 7 | 60.12 | 55 | 148 | 2 | 237.94 | 210 | 713 | 2 | 78,453.16 | 5,810 | 1,557,157 | 66 |
| Scientific Journals Country | WorldBank | R/C (c=2) | DBpedia | 160 | 36.00 | 38 | 53 | 0 | 13.11 | 13 | 23 | 0 | 23.46 | 24 | 28 | 1 | 36.74 | 33 | 94 | 8 | 26.85 | 24 | 162 | 0 | 6,947.59 | 1,440 | 218,957 | 77 |
| | | | YAGO | 150 | 14.09 | 15 | 27 | 0 | 9.44 | 10 | 16 | 0 | 27.80 | 31 | 35 | 0 | 16.48 | 9 | 138 | 5 | 331.16 | 279 | 2,104 | 0 | 17,114.88 | 5,821 | 423,559 | 693 |
| | | | Wikidata | 160 | 4.39 | 4 | 10 | 1 | 4.88 | 5 | 6 | 0 | 123.23 | 119 | 191 | 7 | 60.12 | 55 | 148 | 2 | 237.94 | 210 | 713 | 2 | 78,453.16 | 5,810 | 1,557,157 | 66 |
| Unemployment French Region | SemStats | R/C (c=2) | DBpedia | 26 | 16.38 | 21 | 32 | 0 | 3.73 | 3 | 15 | 0 | 7.81 | 9 | 10 | 3 | 14.19 | 13 | 24 | 7 | 7.19 | 7 | 19 | 1 | 975.88 | 969 | 2,292 | 37 |
| | | | YAGO | 26 | 8.92 | 8 | 14 | 8 | 2.77 | 2 | 8 | 1 | 12.42 | 12 | 14 | 12 | 3.73 | 4 | 6 | 3 | 81.12 | 60 | 299 | 28 | 1,793.19 | 1,424 | 4,527 | 88 |
| | | | Wikidata | 26 | 1.35 | 1 | 3 | 1 | 2.58 | 3 | 4 | 1 | 86.23 | 84 | 119 | 74 | 34.00 | 33 | 51 | 21 | 83.12 | 79 | 157 | 58 | 332.69 | 193 | 1,464 | 137 |
| Endangered Species | a-z-animals | R/C (c=2) | DBpedia | 301 | 11.84 | 12 | 33 | 0 | 6.32 | 5 | 34 | 0 | 10.77 | 11 | 25 | 0 | 2.96 | 3 | 55 | 1 | 12.65 | 11 | 87 | 0 | 566.25 | 15 | 114,742 | 1 |
| | | | YAGO | 65 | 2.48 | 0 | 16 | 0 | 0.76 | 0 | 12 | 0 | 1.78 | 0 | 9 | 0 | 108.62 | 138 | 138 | 1 | 9.53 | 0 | 136 | 0 | 22,286.36 | 28,418 | 28,418 | 1 |
| | | | Wikidata | 301 | 1.05 | 1 | 4 | 0 | 0.44 | 0 | 6 | 0 | 34.32 | 37 | 137 | 3 | 6.94 | 6 | 78 | 0 | 22.04 | 22 | 400 | 1 | 21,909.94 | 70 | 6,460,930 | 0 |
| Drug-Food Interaction | FinkiLOD | C (c=2) | DBpedia | 1,989 | 8.83 | 4 | 38 | 0 | 5.46 | 5 | 18 | 0 | 12.65 | 14 | 15 | 0 | 1.40 | 1 | 5 | 0 | 3.63 | 3 | 12 | 0 | 34.71 | 24 | 158 | 0 |
| | | | YAGO | 588 | 4.46 | 0 | 31 | 0 | 0.68 | 0 | 6 | 0 | 2.15 | 0 | 8 | 0 | 99.69 | 138 | 138 | 1 | 7.28 | 0 | 61 | 0 | 20,427.08 | 28,418 | 28,418 | 1 |
| | | | Wikidata | 1,908 | 1.96 | 2 | 3 | 0 | 0.01 | 0 | 1 | 0 | 45.92 | 47 | 79 | 0 | 2.78 | 2 | 17 | 1 | 34.99 | 27 | 159 | 0 | 32.25 | 26 | 487 | 4 |
| | | | DrugBank | 2,000 | 2.00 | 2 | 2 | 2 | \ | \ | \ | \ | 61.68 | 64 | 71 | 41 | 1.70 | 2 | 2 | 0 | 41.96 | 41 | 132 | 14 | 62.49 | 50 | 211 | 0 |
| | | | FinkiLOD | 2,000 | 1.00 | 1 | 1 | 1 | \ | \ | \ | \ | 3.00 | 3 | 3 | 3 | 0.00 | 0 | 0 | 0 | 1.00 | 1 | 1 | 1 | 0.00 | 0 | 0 | 0 |

Table 4.3: Datasets statistics III

| Dataset | | | types | | | | rel out | | | | rel in | | | | rel-vals out | | | | rel-vals in | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Task | #links | avg | med | max | min | avg | med | max | min | avg | med | max | min | avg | med | max | min | avg | med | max | min |
| AIFB | C (c=4) | 176 | 1.4 | 1 | 2 | 1 | 7.1 | 7 | 9 | 5 | 2.0 | 2 | 5 | 0 | 18.2 | 7 | 219 | 2 | 19.8 | 9 | 246 | 0 |
| AM | C (c=11) | 1,000 | 1.0 | 1 | 1 | 1 | 19.8 | 20 | 29 | 9 | 0.6 | 1 | 3 | 0 | 21.9 | 20 | 283 | 7 | 3.2 | 1 | 273 | 0 |
| MUTAG | C (c=2) | 340 | 1.0 | 1 | 1 | 1 | 9.8 | 10 | 14 | 5 | \ | \ | \ | \ | 65.8 | 56 | 465 | 4 | \ | \ | \ | \ |
| BGS | C (c=2) | 146 | 1.0 | 1 | 1 | 1 | 29.7 | 31 | 36 | 21 | 1.4 | 2 | 4 | 0 | 25.2 | 24 | 54 | 15 | 2.7 | 2 | 12 | 0 |

– The Intersection Tree Path kernel for RDF [58] counts the walks in the subtree that spans from the instance node. Only the walks that go through the instance node are considered. We will therefore refer to it as the root Walk Count (WC) kernel. In terms of performance this kernel is very similar to Intersection SubTree kernels in [170], but it allows for a feature vector representation. The root WC kernel has one parameter: the length of the paths $l$, for which we test 2 and 3.

### 4.2.2 Experiment Setup

To generate features for each dataset we used the latest version of the corresponding LOD dataset to which the dataset is linked. For the datasets that are linked to DBpedia, the features were generated using a local dump of DBpedia 2014 that contains only properties from the `dbpedia-owl` namespace.

We perform two learning tasks, i.e., classification and regression. For classification tasks, we use Naïve Bayes, k-Nearest Neighbors (k=3), C4.5 decision tree, and Support Vector Machines. For the SVM classifier we optimize the parameter $C$ in the range $\{10^{-3}, 10^{-2}, 0.1, 1, 10, 10^2, 10^3\}$. For regression, we use Linear Regression, M5Rules, and k-Nearest Neighbors (k=3). We measure accuracy for classification tasks, and root relative squared error (RRSE) for regression tasks.

The strategies for creating propositional features from Linked Open Data are implemented in the RapidMiner LOD extension[28] [229, 246]. The experiments, including the feature generation and the evaluation, were performed using the RapidMiner data analytics platform[29]. The RapidMiner processes and the complete results can be found online[30]. The experiments were run using a Linux machine with 20GB RAM and 4 Intel Xeon 2.60GHz CPUs.

#### Performance comparison of the strategies:

For both classification and regression, we report the qualitative performance and the runtime for each of the feature generation strategies combined with each learning method. The performances are calculated using stratified 10-fold cross validation. The runtime measures reflect the total time in seconds needed for performing 10-fold cross validation.

One of the most commonly used metrics for comparing the performances of different approaches on a list of dataset is averaging the performances over all datasets. However, if the different datasets are not comparable, as in our case, those averages are only of limited value. Similarly, the use of a paired t-test is not

[28] `http://dws.informatik.uni-mannheim.de/en/research/rapidminer-lod-extension`

[29] `https://rapidminer.com/`

[30] `http://data.dws.informatik.uni-mannheim.de/rmlod/LOD_ML_Datasets/`

advised if the datasets come from different domains and the number of datasets is not too large, as in our case. [60].

Thus, for comparing the approaches, we follow the approach introduced by Demšar [60]. The approach foresees to first rank the strategies for each dataset in isolation, and computing a significance level for the difference of ranks using a Friedman test. Since we have missing values (i.e., not all of the compared approaches were able to process each dataset), we use a variant of the Friedman test, i.e., the Skillings-Mack test [40], which can cope with that problem.

While the Skillings-Mack test only computes whether there is a significant difference between *any* of the compared approaches, pairwise significance levels are computed with a post-hoc Nemenyi test [194]. The results of the post-hoc test allows for concluding if one approach significantly outperforms another one.

In this evaluation we perform the Skillings–Mack with the post-hoc Nemnyi test both on the performances results and the runtimes. We concentrate on analyzing significant differences for each feature propositionalization strategy for the learning methods in isolation. However, with the same methodology, other comparisons (such as comparing the learning methods) would also be possible.

### 4.2.3 Results

Table 4.4 shows the results for the classification tasks, for each strategy and each classification method. The first column for each classification method (#c) shows the number of datasets on which the task was completed, i.e., all the experiments that did not finish in less than ten days, or have run out of memory, were excluded from the final results. The second column shows the macro-average accuracy over all datasets. The third column shows the average accuracy rank, as calculated by the Skillings–Mack test on the accuracy results, where a smaller rank means better performances. The fourth column shows the average runtime over all datasets, and the fifth column shows the average runtime rank, again as calculated by the Skillings-Mack test.

Following [60], to perform the Skillings–Mack test, as we have a total of $N = 19$ datasets and $k = 12$ approaches to compare, we select a significance level of $\alpha = 0.10$, resulting in a critical value $F(11, 198) \approx 1.84$. We carried out the test on each learning method separately. The null hypothesis was rejected for the performances of the strategies when using Naïve Bayes and C4.5, meaning there is a significant performance difference between the strategies. The null hypothesis was rejected for the runtimes of all learning methods, meaning that there is a significant difference in runtimes between the strategies on each method.

For the cases where the null hypothesis was rejected, to detect the pairs of strategies that are significantly different, we performed the post-hoc Nemenyi test, using critical values $q = 0.05$ and $q = 0.10$. The calculated Nemenyi critical difference values are $CD \approx 3.82$ for $q = 0.05$, and $CD \approx 3.54$ for $q = 0.10$. In the table, we show only the comparison of the best strategy to the rest of the strategies (the values marked with $**$ mean that are significantly worse than the

Table 4.4: Classification results, average accuracy, accuracy rank, average runtime (seconds), and runtime rank, for each feature generation strategy, using Naïve Bayes (NB), k-Nearest Neighbors (k-NN, with k=3), C4.5 decision tree (C4.5), and Support Vector Machines (SVM) as classification methods. The best ranked results for each method are marked in bold. The learning models for which the strategies were shown to have significant difference based on the Skillings-Mack test with $\alpha < 0.05$ are marked with *. The single values marked with $**$ mean that are significantly worse than the best strategy at significance level $q = 0.05$, and $*$ for significance level $q = 0.10$.

| Strategy/Method | NB | | | | | k-NN | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | #c | Avg. Acc. | *Acc. Rank | Avg. t | *t Rank | #c | Avg. Acc. | Acc. Rank | Avg. t | *t Rank |
| types | 14 | 0.619 | *7.18 | 2.40 | 3.66 | 14 | 0.677 | 5.45 | 5.23 | 3.29 |
| categories | 14 | 0.668 | 5.39 | 2.60 | 4.32 | 14 | 0.670 | 5.11 | 10.00 | 4.16 |
| rel in | 18 | 0.539 | **7.45 | **0.62** | **2.61** | 18 | 0.634 | 5.63 | **4.56** | **2.66** |
| rel out | 19 | 0.551 | **8.05 | 16.40 | 3.03 | 19 | 0.659 | 6.08 | 30.94 | 3.21 |
| rel in & out | 18 | 0.533 | **7.45 | 3.76 | 3.55 | 18 | 0.678 | 4.87 | 17.98 | 3.79 |
| rel-vals in | 11 | 0.534 | 6.97 | 15.75 | 5.68 | 11 | 0.639 | 6.63 | 54.31 | 5.74 |
| rel-vals out | 19 | 0.722 | 3.53 | 22.42 | 6.11 | 19 | 0.646 | 5.82 | 92.32 | **6.58 |
| rel-vals in & out | 11 | **0.758** | **3.50** | 41.44 | **6.53 | 11 | 0.658 | 5.87 | 150.57 | **6.84 |
| WL_1_2 | 19 | 0.709 | 4.13 | 141.09 | **8.68 | 19 | **0.681** | **4.47** | 446.18 | **7.95 |
| WL_2_2 | 11 | 0.744 | 4.66 | 179.45 | **7.61 | 11 | 0.661 | 5.24 | 313.92 | **7.39 |
| WC_2 | 19 | 0.710 | 4.21 | 58.86 | *6.18 | 19 | 0.673 | 4.79 | 173.63 | 6.18 |
| WC_3 | 18 | 0.721 | 4.11 | 155.54 | **8.05 | 18 | 0.627 | 6.68 | 341.10 | **8.21 |

| Strategy/Method | C4.5 | | | | | SVM | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | #c | Avg. Acc. | *Acc. Rank | Avg. t | *t Rank | #c | Avg. Acc. | Acc. Rank | Avg. t | *t Rank |
| types | 14 | 0.712 | 4.78 | 97.24 | 3.68 | 14 | 0.734 | 4.71 | 139.99 | 3.53 |
| categories | 14 | 0.723 | 5.25 | 234.63 | 4.26 | 14 | 0.730 | 5.00 | **119.20** | 4.21 |
| rel in | 18 | 0.664 | 5.72 | **27.53** | **2.76** | 18 | 0.688 | 5.66 | 223.15 | **2.66** |
| rel out | 19 | 0.702 | 6.03 | 134.10 | 3.79 | 19 | 0.689 | 5.76 | 574.29 | 3.89 |
| rel in & out | 18 | 0.706 | 5.19 | 86.26 | 4.08 | 18 | 0.725 | 5.18 | 581.19 | 4.13 |
| rel-vals in | 10 | 0.641 | 6.06 | 127.46 | 4.55 | 11 | 0.680 | 6.00 | 1,004.73 | 4.58 |
| rel-vals out | 15 | 0.750 | 5.14 | 286.96 | 5.68 | 19 | 0.778 | 5.00 | 954.21 | 5.74 |
| rel-vals in & out | 6 | 0.837 | 4.56 | 493.89 | 5.34 | 11 | **0.821** | 4.53 | 1,737.55 | 5.37 |
| WL_1_2 | 19 | 0.737 | 4.08 | 2,515.64 | **7.61 | 19 | 0.777 | 4.50 | 835.24 | **7.66 |
| WL_2_2 | 5 | **0.875** | **3.94** | 945.11 | 5.42 | 11 | 0.800 | **4.03** | 1,068.76 | 5.68 |
| WC_2 | 18 | 0.749 | 4.36 | 1,656.64 | 5.55 | 19 | 0.772 | 4.61 | 638.60 | 5.55 |
| WC_3 | 14 | 0.761 | 4.89 | 1,091.78 | **6.63 | 18 | 0.782 | 5.03 | 1,585.24 | **6.68 |

best strategy at significance level $q = 0.05$, and $*$ for significance level $q = 0.10$). The complete pairwise results can be found online.

From the results, we can observe that only for Naïve Bayes, there are significant differences w.r.t. accuracy based on the Nemenyi post-hoc test. Although we rejected the null hypothesis for C4.5 based on the Skillings–Mack test, the Nemenyi post-hoc test is not powerful enough to detect any significant differences between the strategies, on the significance levels we chose. Furthermore, for the SVM experiments, we can observe that there is a disagreement for the best strategy based on the best average accuracy and the average rank. These findings show that only comparing average accuracies is not enough for making significant statements about differences of approaches.

Table 4.5 shows the results for the regression tasks, for each strategy and each regression method. Like for classification, the first column for each regression method (#c) shows the number of datasets on which the task was completed. The second column shows the average *root relative squared error* (RRSE) over all datasets. The third column shows the average RRSE rank, as calculated by the Skillings–Mack test on the RRSE results, where smaller rank means better performances. The fourth column shows the average runtime over all datasets, and the fifth column shows the average runtime rank.

Again following [60], as we have $N = 15$ datasets and $k = 12$ approaches to compare, we select the significance level $\alpha = 0.10$, resulting in a critical value $F(11, 154) \approx 1.85$. We carried out the test on each learning method separately. The null hypothesis was rejected for the performances of the strategies when using Linear Regression and k-NN methods, meaning there is a significant performance difference between the strategies. The null hypothesis was rejected for the runtimes of all learning methods, meaning that there are significant differences in runtimes between the strategies on each learning method.

For the cases where the null hypothesis was rejected, to detect the pairs of strategies that are significantly different, we performed the post-hoc Nemenyi test. The calculated Nemenyi critical difference values are $CD \approx 4.3$ for $q = 0.05$, and $CD \approx 3.99$ for $q = 0.10$.Although we rejected the null hypothesis for the RRSE results for all learning methods, the Nemenyi test is not powerful enough to detect any significant differences between the strategies.

Like for classification, we can observe that there is a disagreement for the best strategy based on the best average RRSE, and the average rank for the performances and the runtimes for all learning models.

### 4.2.4 Number of Generated Features

In this section, we compare the number of features generated by the different feature generation strategies, since that number has a direct influence on performance and memory consumption of the learning step. Like for performance and runtime, we use the Skillings–Mack test.

To perform the Skillings–Mack test, we have $N = 21$ datasets and $k = 12$

Table 4.5: Regression results, average RRSE, RRSE rank, average runtime (seconds), and runtime rank, for each feature generation strategy, using Linear Regression (LR), M5Rules (M5), and k-Nearest Neighbors(k-NN, with k=3) as regression methods. The best ranked results for each method are marked in bold. The learning models for which the strategies were shown to have significant difference based on the Skillings-Mack test with $\alpha < 0.05$ are marked with *. The single values marked with $**$ mean that are significantly worse than the best strategy at significance level $q = 0.05$, and $*$ for significance level $q = 0.10$.

| Strategy/Method | LR | | | | |
|---|---|---|---|---|---|
| | #c | Avg. RRSE | *RRSE Rank | Avg. t | *t Rank |
| types | 15 | 1.331 | 6.53 | 21.32 | **2.13** |
| categories | 15 | 0.919 | 4.13 | 170.36 | 3.53 |
| rel in | 15 | 0.706 | 4.77 | 3.90 | 2.40 |
| rel out | 15 | 0.800 | 4.83 | **1.94** | 2.67 |
| rel in & out | 15 | 0.732 | 5.20 | 15.07 | 4.40 |
| rel-vals in | 2 | 0.295 | 3.97 | 2,497.38 | 5.40 |
| rel-vals out | 9 | 0.575 | 3.97 | 589.97 | 5.93 |
| rel-vals in & out | 2 | **0.287** | 3.97 | 1,253.62 | 5.27 |
| WL_1_2 | 9 | 0.631 | 4.17 | 1,591.99 | *6.17 |
| WL_2_2 | 3 | 0.383 | 3.84 | 2,092.63 | 5.27 |
| WC_2 | 10 | 0.570 | 4.40 | 1,046.98 | 5.27 |
| WC_3 | 9 | 0.631 | **3.83** | 2,221.45 | *6.77 |
| Strategy/Method | k-NN | | | | |
| | #c | Avg. RRSE | *RRSE Rank | Avg. t | *t Rank |
| types | 15 | 0.967 | 7.40 | 3.25 | **2.27** |
| categories | 15 | 0.751 | 5.87 | 10.57 | 3.73 |
| rel in | 15 | 1.005 | 6.57 | **0.64** | 2.43 |
| rel out | 15 | 0.756 | 5.77 | 0.94 | 2.60 |
| rel in & out | 15 | 0.728 | 5.07 | 11.29 | 4.63 |
| rel-vals in | 9 | 1.533 | 8.07 | 30.20 | *6.27 |
| rel-vals out | 15 | 0.713 | 5.27 | 81.70 | **7.53 |
| rel-vals in & out | 9 | 0.791 | 5.53 | 180.90 | **8.27 |
| WL_1_2 | 15 | 0.721 | 5.23 | 172.37 | **8.57 |
| WL_2_2 | 9 | **0.611** | 5.27 | 22.72 | **8.20 |
| WC_2 | 15 | 0.743 | **4.37** | 45.80 | **6.67 |
| WC_3 | 15 | 0.956 | 6.40 | 335.10 | **9.63 |
| Strategy/Method | M5 | | | | |
| | #c | Avg. RRSE | *RRSE Rank | Avg. t | *t Rank |
| types | 15 | 0.671 | 5.77 | 21.23 | **2.47** |
| categories | 15 | 0.668 | 4.83 | 39.14 | 3.33 |
| rel in | 15 | 0.699 | 6.30 | **10.13** | 2.53 |
| rel out | 15 | 0.715 | 7.37 | 18.33 | 2.93 |
| rel in & out | 15 | 0.754 | 7.23 | 90.60 | 5.07 |
| rel-vals in | 9 | 0.676 | 6.97 | 194.59 | 6.20 |
| rel-vals out | 15 | 0.671 | 5.60 | 419.69 | **6.87 |
| rel-vals in & out | 9 | **0.598** | 6.00 | 1,688.85 | **8.07 |
| WL_1_2 | 15 | 0.658 | 6.00 | 1,912.98 | **8.33 |
| WL_2_2 | 9 | 0.607 | 4.60 | 262.98 | **7.93 |
| WC_2 | 15 | 0.666 | 6.10 | 588.32 | **7.37 |
| WC_3 | 15 | 0.618 | **4.03** | 3,352.31 | **9.70 |

Table 4.6: Number of generated features

| Strategy | #c | Avg. | *Rank |
|---|---|---|---|
| types | 16 | 1258.13 | 3.12 |
| categories | 16 | 2407.31 | 4.12 |
| rel in | 20 | **615.43** | **2.12** |
| rel out | 21 | 646.65 | 2.50 |
| rel in & out | 20 | 1224.19 | 3.83 |
| rel-vals in | 20 | 180903.80 | **7.81 |
| rel-vals out | 21 | 17099.25 | **6.83 |
| rel-vals in & out | 20 | 181781.52 | **9.48 |
| WL_1_2 | 21 | 29642.19 | **8.43 |
| WL_2_2 | 21 | 166769.33 | **10.67 |
| WC_2 | 21 | 11116.90 | **6.21 |
| WC_3 | 21 | 53248.29 | **9.45 |

approaches, we select significance level $\alpha = 0.10$, resulting in a critical value $F(11, 220) \approx 1.83$. The observed value is $F_F \approx 12.4$ is higher than the critical value, so the null hypothesis is rejected at a $10\%$ significance level. Furhtermore, we performed the post-hoc Nemenyi test, using critical value $q = 0.05$ and $q = 0.10$, to make pairwise comparisons. The calculated Nemenyi critical difference values are $CD \approx 3.63$ for $q = 0.05$, and $CD \approx 3.37$ for $q = 0.10$.

Table 4.6 shows only the comparison of the best strategy to the rest of the strategies (the values marked with $**$ mean that are significantly worse than the best strategy at significance level $q = 0.05$, and $*$ for significance level $q = 0.10$). The first column shows the number of datasets for which the feature set was generated. The second column shows the average of generated features over all datasets, and the third column is the average ranked calculated using the Skillings–Mack approach.

The results show that the strategy *rel in* generates the lowest number of features for each dataset, and that the strategies based on values, as well as the kernels, create significantly more features.

### 4.2.5 Features Increase Rate

Finally, we conduct a scalability experiment, where we examine how the number of instances affects the number of generated features by each feature generation strategy. For this purpose we use the *Metacritic Movies* dataset. We start with a random sample of 100 instances, and in each next step we add 200 (or 300) unused instances, until the complete dataset is used, i.e., $2,000$ instances. The number of generated features for each sub-sample of the dataset using each of the feature generation strategies is shown in Figure 8.3. We can observe that in the beginning the curves for all strategies sharply increase. After the sub-sample reaches the

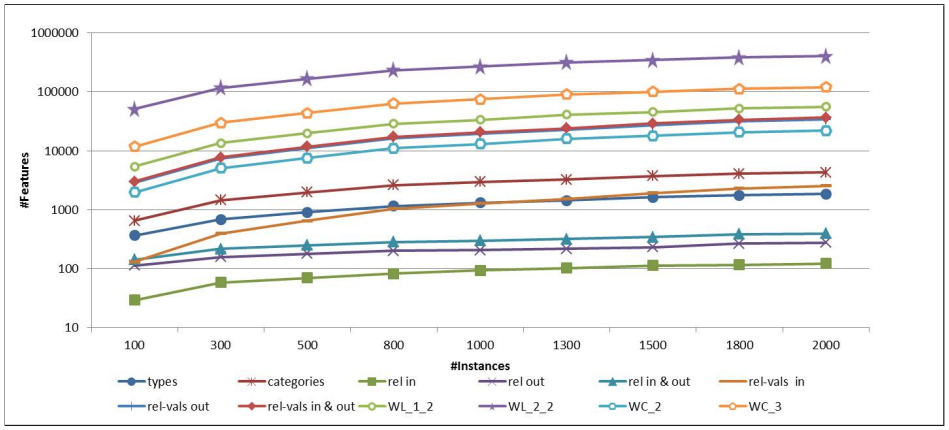



Figure 4.1: Features increase rate per strategy (log scale).

half of the complete sample, the strategies based on generic relations stabilize, as only a few new relations are discovered when adding new instances. The curves for the strategies based on generic relation-values, and the specific relations, are steadily increasing as new instances are added. On the other side, the curves for the strategies based on graph substructures increase more rapidly than the rest of the strategies, without a sign of convergence. From the chart, we can also observe that the strategies based on graph substructures generate feature sets three orders of magnitude large than the strategies based on generic relations, and two orders of magnitude larger than the strategies based on specific relations.

## 4.3 Conclusion and Outlook

In this chapter, we have introduced a collection of 22 benchmark datasets for machine learning on the Semantic Web. We have shown how they can be used to set up experiments which allow for making statistically significant comparisons between different learning approaches. So far, we have concentrated on classification and regression tasks. There are methods to derive clustering and outlier detection benchmarks from classification and regression datasets [77, 84], so that extending the dataset collection for such unsupervised tasks is possible as well. Furthermore, as many datasets on the Semantic Web use extensive hierarchies in the form of ontologies, building benchmark datasets for tasks like *hierarchical multi-label classification* [279] would also be an interesting extension.

At the moment, the dataset collection has a certain bias towards datasets linked to DBpedia. This has two main reasons, (1) DBpedia being a cross-domain knowledge base usable in datasets from very different topical domains, and (2) tools like DBpedia Lookup and DBpedia Spotlight making it easy to link external datasets to DBpedia. However, DBpedia can be seen as an entry point to the Web of Linked Data, with many datasets linking to and from DBpedia. In fact, some Semantic

Web mining tools, such as the RapidMiner Linked Open Data extension, are capable of exploiting such links automatically and combining information from various Linked Data sets [246].

Summarizing, this presents the first attempt of creating a universal benchmark collection for Semantic Web mining, an area in which much research is conducted, but an accepted benchmark set is missing. By successively extending this benchmark set, we believe that it will provide a useful cornerstone for research at the crossroads of Semantic Web and machine learning.

# Chapter 5

# Propositionalization Strategies for Creating Features from Linked Open Data

As shown in chapter 2, Semantic Web knowledge graphs have been recognized as a valuable source of background knowledge in many data mining tasks. Augmenting a dataset with features taken from Semantic Web knowledge graphs can, in many cases, improve the results of a data mining problem at hand, while externalizing the cost of maintaining that background knowledge [221].

Most data mining algorithms work with a propositional *feature vector* representation of the data, i.e., each instance is represented as a vector of features $\langle f_1, f_2, ..., f_n \rangle$, where the features are either binary (i.e., $f_i \in \{true, false\}$), numerical (i.e., $f_i \in \mathbb{R}$), or nominal (i.e., $f_i \in S$, where $S$ is a finite set of symbols). Linked Open Data, however, comes in the form of *graphs*, connecting resources with types and relations, backed by a schema or ontology.

Thus, for accessing Semantic Web knowledge graphs with existing data mining tools, transformations have to be performed, which create propositional features from the graphs in Linked Open Data, i.e., a process called *propositionalization* [154]. Usually, binary features (e.g., `true` if a type or relation exists, `false` otherwise) or numerical features (e.g., counting the number of relations of a certain type) are used [225]. Other variants, e.g., computing the fraction of relations of a certain type, are possible, but rarely used.

Our hypothesis is that the strategy of creating propositional features from Linked Open Data may have an influence on the data mining result. For example, promiximity-based algorithms like k-NN will behave differently depending on the strategy used to create numerical features, as that strategy has a direct influence on most distance functions.

In this chapter, we compare a set of different strategies for creating features from types and relations in Linked Open Data. We compare those strategies on a number of different datasets and across different tasks, i.e., classification, regres-

sion, and outlier detection.

**The work presented in this chapter has been published before as: "Petar Ristoski, Heiko Paulheim: *Feature selection in hierarchical feature spaces.* Proceedings of the 17th International Conference on Discovery Science, Bled, Slovenia, October, 2014." [253].**

## 5.1 Strategies

When creating features for a resource, we take into account the relation to other resources. We distinguish strategies that use the object of *specific relations*, and strategies that only take into account the presence of *relations as such*.

### 5.1.1 Strategies for Features Derived from Specific Relations

Some relations in Linked Open Data sources play a specific role. One example are `rdf:type` relations assigning a direct type to a resource. A statement `r rdf:type C` is typically translated into description logics as $C(r)$, i.e., `rdf:type` is treated differently from any other predicate. For some datasets, similar relations exist, e.g., the `dcterms:subject` relations in DBpedia [162] which contain a link to the category of the original Wikipedia article a DBpedia resource is derived from.

For such relations, we propose three strategies:

- Creating a *binary feature* indicating presence or absence of the relation's object.
- Creating a *relative count feature* indicating the relative count of the relation's object. For a resource that has a relation to $n$ objects, each feature value is $\frac{1}{n}$.
- Creating a *TF-IDF feature*, whose value is $\frac{1}{n} \cdot log\frac{N}{|\{r|C(r)\}|}$, where $N$ is the total number of resources in the dataset, and $|\{r|C(r)\}|$ denotes the number of resources that have the respective relation $r$ to $C$.

The rationale for using relative counts is that if there are only a few relations of a particular kind, each individual related object may be more important. For example, for a general book which has a hundred topics, each of those topics is less characteristic for the book than a specific book with only a few topics. Thus, that strategy takes into account both the existence and the importance of a certain relation.

The rationale for using TF-IDF is to further reduce the influence of too general features, in particular when using a distance-based mining algorithm. Table 5.1 shows the features generated for the example depicted in Fig.5.1. It can be observed that using TF-IDF implicitly gives a higher weight to more specific features, which can be important in distance-based mining algorithms (i.e., it increases the similarity of two objects more if they share a more specific type than a more abstract one).

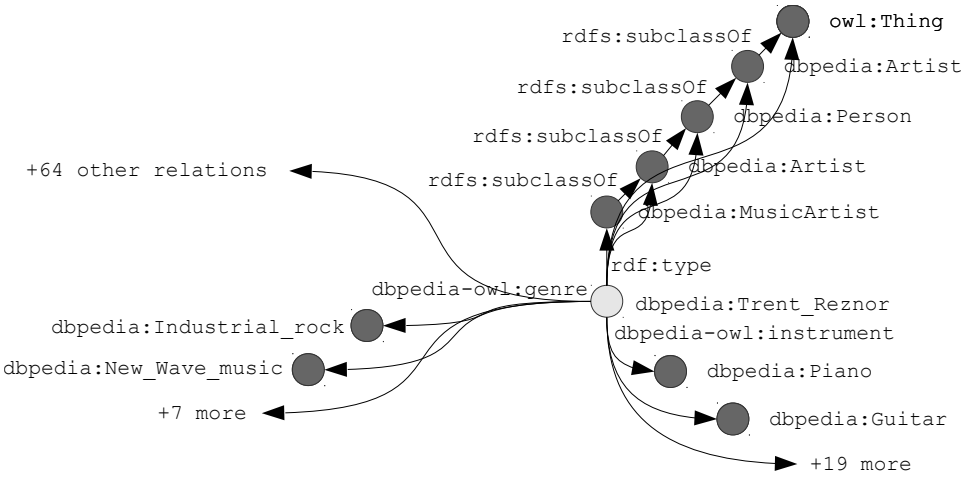



Figure 5.1: Example DBpedia resource (`dbpedia:Trent_Reznor`) and an excerpt of its types and relations

Table 5.1: Features for `rdf:type` and relations as such, generated for the example shown in Fig. 5.1. For TF-IDF, we assume that there are 1,000 instances in the dataset, all of which are persons, 500 of which are artists, and 100 of which are music artists with genres and instruments.

| | Specific relation: `rdf:type` | | | | | Relations as such | |
|---|---|---|---|---|---|---|---|
| Strategy | `MusicArtist` | `Artist` | `Person` | `Agent` | `Thing` | `genre` | `instrument` |
| Binary | `true` | `true` | `true` | `true` | `true` | `true` | `true` |
| Count | – | – | – | – | – | 9 | 21 |
| Relative Count | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.091 | 0.212 |
| TF-IDF | 0.461 | 0.139 | 0 | 0 | 0 | 0.209 | 0.488 |

### 5.1.2 Strategies for Features Derived from Relations as Such

Generic relations describe how resources are related to other resources. For example, a writer is connected to her birthplace, her alma mater, and the books she has written. Such relations between a resource $r$ and a resource $r'$ are expressed in description logics as $p(r, r')$ (for an outgoing relation) or $p(r', r)$ (for an incoming relation), where $p$ can be any relation.

In general, we treat incoming (rel in) and outgoing (rel out) relations. For such generic relations, we propose four strategies:

- Creating a *binary feature* for each relation.
- Creating a *count feature* for each relation, specifying the number of resources connected by this relation.
- Creating a *relative count feature* for each relation, specifying the fraction of resources connected by this relation. For a resource that has total number of $P$ outgoing relations, the relative count value for a relation $p(r, r')$ is defined as $\frac{n_p}{P}$,

where $n_p$ is the number of outgoing relations of type $p$. The feature is defined accordingly for incoming relations

- Creating a *TF-IDF feature* for each relation, whose value is $\frac{n_p}{P} \cdot log \frac{N}{|\{r|\exists r' : p(r,r')\}|}$, where $N$ is the overall number of resources, and $|\{r|\exists r' : p(r, r')\}|$ denotes the number of resources for which the relation $p(r, r')$ exists. The feature is defined accordingly for incoming relations.

The rationale of using relative counts is that resources may have multiple types of connections to other entities, but not all of them are equally important. For example, a person who is mainly a musician may also have written one book, but recorded many records, so that the relations get different weights. In that case, he will be more similar to other musicians than to other authors – which is not the case if binary features are used.

The rationale of using TF-IDF again is to reduce the influence of too general relations. For example, two persons will be more similar if both of them have recorded records, rather than if both have a last name. The IDF factor accounts for that weighting. Table 5.1 shows the features generated from the example in Fig. 5.1.

## 5.2 Evaluation

We evaluated the strategies outlined above on six different datasets, two for each task of classification, regression, and outlier detection.

### 5.2.1 Tasks and Datasets

The following datasets were used in the evaluation:

- The *Auto MPG* data set[1], a dataset that captures different characteristics of cars (such as cyclinders, transmission horsepower), and the target is to predict the fuel consumption in Miles per Gallon (MPG) as a regression task [243]. Each car in the dataset was linked to the corresponding resource in DBpedia.
- The *Cities* dataset contains a list of cities and their quality of living (as a numerical score), as captured by Mercer [219]. The cities are mapped to DBpedia. We use the dataset both for regression as well as for classification, discretizing the target variable into high, medium, and low.
- The *Sports Tweets* dataset consists of a number of tweets, with the target class being whether the tweet is related to sports or not.[2] The dataset was mapped to DBpedia using DBpedia Spotlight [174].

[1] `http://archive.ics.uci.edu/ml/datasets/Auto+MPG`

[2] `https://github.com/vinaykola/twitter-topic-classifier/blob/master/training.txt`

Table 5.2: Datasets used in the evaluation. Tasks: C=Classification, R=Regression, O=Outlier Detection

| Dataset | Task | # instances | # types | # categories | # rel in | # rel out | # rel in & out |
|---|---|---|---|---|---|---|---|
| Auto MPG | R | 391 | 264 | 308 | 227 | 370 | 597 |
| Cities | C/R | 212 | 721 | 999 | 1,304 | 1,081 | 2,385 |
| Sports Tweets | C | 5,054 | 7,814 | 14,025 | 3,574 | 5,334 | 8,908 |
| DBpedia-Peel | O | 2,083 | 39 | - | 586 | 322 | 908 |
| DBpedia-DBTropes | O | 4,228 | 128 | - | 912 | 2,155 | 3,067 |

- The *DBpedia-Peel* dataset is a dataset where each instance is a link between the DBpedia and the Peel Sessions LOD datasets. Outlier detection is used to identify links whose characteristics deviate from the majority of links, which are then regarded to be wrong. A partial gold standard of 100 links exists, which were manually annotated as right or wrong [222].

- The *DBpedia-DBTropes* dataset is a similar dataset with links between DBpedia and DBTropes.

For the classification and regression tasks, we use direct types (i.e., `rdf:type`) and DBpedia categories (i.e., `dcterms:subject`), as well as all strategies for generic relations. For the outlier detection tasks, we only use direct types and generic relations, since categories do not exist in the other LOD sources involved. An overview of the datasets, as well as the size of each feature set, is given in Table 5.2.

For classification tasks, we use Naïve Bayes, k-Nearest Neighbors (with k=3), and C4.5 decision tree. For regression, we use Linear Regression, M5Rules, and k-Nearest Neighbors (with k=3). For outlier detection, we use Global Anomaly Score (GAS, with k=25), Local Outlier Factor (LOF), and Local Outlier Probabilities (LoOP, with k=25). We measure accuracy for classification tasks, root-mean-square error (RMSE) for regression tasks, and area under the ROC curve (AUC) for outlier detection tasks.

The evaluations are performed in RapidMiner, using the Linked Open Data extension [229]. For classification, regression, and outlier detection, we use the implementation in RapidMiner where available, otherwise, the corresponding implementations from the Weka[3] and Anomaly Detection [100] extension in RapidMiner were used. The RapidMiner processes and datasets used for the evaluation can be found online.[4] The strategies for creating propositional features from Linked Open Data are implemented in the RapidMiner Linked Open Data extension[5] [229].

[3] `https://marketplace.rapid-i.com/UpdateServer/faces/product_details.xhtml?productId=rmx_weka`

[4] `http://data.dws.informatik.uni-mannheim.de/propositionalization_strategies/`

[5] `http://dws.informatik.uni-mannheim.de/en/research/rapidminer-lod-extension`

Table 5.3: Classification accuracy results for the Cities and Sports Tweets datasets, using Naïve Bayes(NB), k-Nearest Neighbors (k-NN, with k=3), and C4.5 decision tree (C4.5) as classification algorithms, on five different feature sets, generated using three propositionalization strategies, for *types* and *categories* feature sets, and four propositionalization strategies for the *incoming* and *outgoing relations* feature sets. The best result for each feature set, for each classification algorithm is marked in bold.

| Datasets | | Cities | | | | Sports Tweets | | | |
|---|---|---|---|---|---|---|---|---|---|
| Features | Representation | NB | k-NN | C4.5 | Avg. | NB | k-NN | C4.5 | Avg. |
| types | Binary | .557 | **.561** | **.590** | **.569** | .8100 | **.829** | **.829** | .822 |
| | Relative Count | **.571** | .496 | .552 | .539 | .809 | .814 | .818 | .814 |
| | TF-IDF | **.571** | .487 | .547 | .535 | **.821** | .824 | .826 | **.824** |
| categories | Binary | .557 | **.499** | .561 | .539 | .822 | .765 | .719 | .769 |
| | Relative Count | **.595** | .443 | **.589** | **.542** | **.907** | **.840** | .808 | **.852** |
| | TF-IDF | .557 | **.499** | .570 | .542 | .896 | .819 | **.816** | .844 |
| rel in | Binary | **.604** | **.584** | **.603** | **.597** | .831 | .836 | .846 | .838 |
| | Count | .566 | .311 | .593 | .490 | **.832** | **.851** | **.854** | **.845** |
| | Relative Count | .491 | .382 | .585 | .486 | .695 | .846 | .851 | .7977 |
| | TF-IDF | .349 | .382 | .542 | .424 | .726 | .846 | .849 | .8077 |
| rel out | Binary | .476 | **.600** | .567 | **.547** | **.806** | .823 | .844 | .824 |
| | Count | **.499** | .552 | **.585** | .546 | .799 | .833 | **.850** | **.827** |
| | Relative Count | .480 | .584 | .566 | .543 | .621 | .842 | .835 | .766 |
| | TF-IDF | .401 | .547 | .585 | .511 | .699 | **.844** | .841 | .7949 |
| rel in & out | Binary | **.594** | **.585** | .564 | **.581** | **.861** | .851 | .864 | .859 |
| | Count | .561 | .542 | **.608** | .570 | .860 | **.860** | **.871** | **.864** |
| | Relative Count | .576 | .471 | .565 | .537 | .700 | .845 | .872 | .8058 |
| | TF-IDF | .401 | .462 | .584 | .482 | .751 | .848 | .861 | .820 |

### 5.2.2 Results

For each of the three tasks we report the results for each of the feature sets, generated using different propositionalization strategies. The classification and regression results are calculated using stratified 10-fold cross validation, while for the outlier detection the evaluations were made on the partial gold standard of 100 links for each of the datasets.[6]

Table 5.3 shows the classification accuracy for the Cities and Sports Tweets datasets. We can observe that the results are not consistent, but the best results for each classifier and for each feature set are achieved using different representation strategy. Only for the incoming relations feature set, the best results for the Cities dataset for each classifier are achieved when using the *Binary* strategy, while for the Sports Tweets dataset the best results are achieved when using *Count* strategy. We can observe that for most of the generic relation feature sets using *TF-IDF* strategy leads to poor results. That can be explained with the fact that *TF-IDF* tends to give higher weights to relations that appear rarely in the dataset, which also might be a result of erroneous data. Also, on the Cities dataset it can be noticed that when using k-NN on the incoming relations feature set, the difference in the results using different strategies is rather high.

Table 5.4 shows the results of the regression task for the Auto MPG and Cities

[6]Note that we measure the capability of finding errors by outlier detection, not of outlier detection as such, i.e., natural outliers may be counted as false positives.

datasets. For the Auto MPG dataset, for M5Rules and k-NN classifiers the best results are achieved when using *Relative Count* and *TF-IDF* for all feature sets, while the results for LR are mixed. For the Cities dataset we can observe that the results are mixed for the types and categories feature set, but for the generic relations feature sets, the best results are achieved when using *Binary* representation. Also, it can be noticed that when using linear regression, there is a drastic difference in the results between the strategies.

Table 5.5 shows the results of the outlier detection task for the DBpedia-Peel and DBpedia-DBTropes datasets. In this task we can observe much higher difference in performances when using different propositionalization strategies. We can observe that the best results are achieved when using relative count features. The explanation is that in this task, we look at the implicit types of entities linked when searching for errors (e.g., a book linked to a movie of the same name), and those types are best characterized by the distribution of relations, as also reported in [224]. On the other hand, TF-IDF again has the tendency to assign high weights to rare features, which may also be an effect of noise.

By analyzing the results on each task, we can conclude that the chosen propositionalization strategy has major impact on the overall results. Also, in some cases there is a drastic performance differences between the strategies that are used. Therefore, in order to achieve the best performances, it is important to choose the most suitable propositionalization strategy, which mainly depends on the given dataset, the given data mining task, and the data mining algorithm to be used.

When looking at aggregated results, we can see that for the classification and regression tasks, binary and count features work best in most cases. Furthermore, we can observe that algorithms that rely on the concept of *distance*, such as k-NN, linear regression, and most outlier detection methods, show a stronger variation of the results across the different strategies than algorithms that do not use distances (such as decision trees).

## 5.3 Conclusion and Outlook

Until now, the problem of finding the most suitable propositionalization strategy for creating features from Semantic Web knowledge graphs has not been tackled, as previous researches focused only on binary, or in some cases numerical representation of features. In this chapter, we have compared different strategies for creating propositional features from types and relations in Linked Open Data. We have implemented three propositionalization strategies for specific relations, like `rdf:type` and `dcterms:subject`, and four strategies for generic relations. We conducted experiments on six different datasets, across three different data mining tasks, i.e. classification, regression and outlier detection. The experiments show that the chosen propositionalization strategy might have a major impact on the overall results. However, it is difficult to come up with a general recommendation for a strategy, as it depends on the given data mining task, the given dataset,

Table 5.4: Root-mean-square error (RMSE) results for the Auto MPG and Cities datasets, using Linear Regression (LR), M5Rules (M5), and k-Nearest Neighbors(k-NN, with k=3) as regression algorithms, on five different feature sets, generated using three propositionalization strategies, for *types* and *categories* feature sets, and four propositionalization strategies for the *incoming* and *outgoing relations* feature sets. The best result for each feature set, for each regression algorithm is marked in bold.

| Datasets | | Auto MPG | | | | Cities | | | |
|---|---|---|---|---|---|---|---|---|---|
| Features | Representation | LR | M5 | k-NN | Avg. | LR | M5 | k-NN | Avg. |
| types | Binary | 3.95 | 3.05 | 3.63 | 3.54 | 24.30 | 18.79 | **22.16** | 21.75 |
| | Relative Count | **3.84** | **2.95** | **3.57** | **3.45** | 18.04 | 19.69 | 33.56 | 23.77 |
| | TF-IDF | 3.86 | 2.96 | **3.57** | 3.46 | **17.85** | **18.77** | 22.39 | **19.67** |
| categories | Binary | **3.69** | 2.90 | 3.61 | **3.40** | **18.88** | 22.32 | **22.67** | **21.29** |
| | Relative Count | 3.74 | 2.97 | 3.57 | 3.43 | 18.95 | **19.98** | 34.48 | 24.47 |
| | TF-IDF | 3.78 | **2.90** | **3.56** | 3.41 | 19.02 | 22.32 | 23.18 | 21.51 |
| rel in | Binary | **3.84** | 2.86 | 3.61 | **3.44** | **49.86** | **19.20** | **18.53** | **29.20** |
| | Count | 3.89 | 2.96 | 4.61 | 3.82 | 138.04 | 19.91 | 19.2 | 59.075 |
| | Relative Count | 3.97 | 2.91 | **3.57** | 3.48 | 122.36 | 22.33 | 18.87 | 54.52 |
| | TF-IDF | 4.10 | **2.84** | **3.57** | 3.50 | 122.92 | 21.94 | 18.56 | 54.47 |
| rel out | Binary | **3.79** | 3.08 | 3.59 | **3.49** | **20.00** | **19.36** | **20.91** | **20.09** |
| | Count | 4.07 | 2.98 | 4.14 | 3.73 | 36.31 | 19.45 | 23.99 | 26.59 |
| | Relative Count | 4.09 | **2.94** | **3.57** | 3.53 | 43.20 | 21.96 | 21.47 | 28.88 |
| | TF-IDF | 4.13 | 3.00 | 3.57 | 3.57 | 28.84 | 20.85 | 22.21 | 23.97 |
| rel in & out | Binary | 3.99 | 3.05 | 3.67 | 3.57 | **40.80** | **18.80** | **18.21** | **25.93** |
| | Count | 3.99 | 3.07 | 4.54 | 3.87 | 107.25 | 19.52 | 18.90 | 48.56 |
| | Relative Count | **3.92** | **2.98** | **3.57** | **3.49** | 103.10 | 22.09 | 19.60 | 48.26 |
| | TF-IDF | 3.98 | 3.01 | **3.57** | 3.52 | 115.37 | 20.62 | 19.70 | 51.89 |

and the data mining algorithm to be used.

For future work, additional experiments can be performed on more feature sets. For example, a feature sets of qualified incoming and outgoing relation can be generated, where qualified relations attributes beside the type of the relation take the type of the related resource into account. The evaluation can be extended on more datasets, using and combining attributes from multiple Linked Open Data sources. Also, it may be interesting to examine the impact of the propositionalization strategies on even more data mining tasks, such as clustering and recommender systems.

So far, we have considered only statistical measures for feature representation without exploiting the semantics of the data. More sophisticated strategies that combine statistical measures with the semantics of the data can be developed. For example, we can represent the connection between different resources in the graph by using some of the standard properties of the graph, such as the depth of the hierarchy level of the resources, the fan-in and fan-out values of the resources, etc.

The problem of propositionalization and feature weighting has been extensively studied in the area of text categorization [61, 155]. Many approaches have been proposed, which can be adapted and applied on Linked Open Data datasets. For example, adapting supervised weighting approaches, such as [98, 283], might

Table 5.5: Area under the ROC curve (AUC) results for the DBpedia-Peel and Dbpedia-DBTropes datasets, using Global Anomaly Score (GAS, with k=25), Local Outlier Factor (LOF), and Local Outlier Probabilities (LoOP, with k=25) as outlier detection algorithms, on four different feature sets, generated using three propositionalization strategies, for *types* feature set, and four propositionalization strategies for the *incoming* and *outgoing relations* feature sets. The best result for each feature set, for each outlier detection algorithm is marked in bold.

| Datasets | | DBpedia-Peel | | | | DBpedia-DBTropes | | | |
|---|---|---|---|---|---|---|---|---|---|
| Features | Representation | GAS | LOF | LoOP | Avg. | GAS | LOF | LoOP | Avg. |
| types | Binary | **0.386** | 0.486 | 0.554 | 0.476 | **0.503** | 0.627 | **0.605** | **0.578** |
| | Relative Count | 0.385 | 0.398 | 0.595 | 0.459 | **0.503** | 0.385 | 0.314 | 0.401 |
| | TF-IDF | **0.386** | **0.504** | **0.602** | **0.497** | **0.503** | **0.672** | 0.417 | 0.531 |
| rel in | Binary | 0.169 | 0.367 | 0.288 | 0.275 | 0.425 | 0.520 | 0.450 | 0.465 |
| | Count | 0.200 | 0.285 | 0.290 | 0.258 | 0.503 | **0.590** | **0.602** | **0.565** |
| | Relative Count | **0.293** | **0.496** | **0.452** | **0.414** | **0.589** | 0.555 | 0.493 | 0.546 |
| | TF-IDF | 0.140 | 0.353 | 0.317 | 0.270 | 0.509 | 0.519 | 0.568 | 0.532 |
| rel out | Binary | 0.250 | 0.195 | 0.207 | 0.217 | 0.325 | 0.438 | 0.432 | 0.398 |
| | Count | 0.539 | 0.455 | **0.391** | 0.462 | 0.547 | 0.577 | **0.522** | 0.549 |
| | Relative Count | **0.542** | **0.544** | **0.391** | **0.492** | **0.618** | 0.601 | 0.513 | **0.577** |
| | TF-IDF | 0.116 | 0.396 | 0.240 | 0.251 | 0.322 | **0.629** | 0.471 | 0.474 |
| rel in & out | Binary | 0.324 | 0.430 | 0.510 | 0.422 | 0.351 | 0.439 | 0.396 | 0.396 |
| | Count | 0.527 | 0.367 | 0.454 | 0.450 | 0.565 | 0.563 | 0.527 | 0.553 |
| | Relative Count | **0.603** | **0.744** | **0.616** | **0.654** | **0.667** | **0.672** | **0.657** | **0.665** |
| | TF-IDF | 0.202 | 0.667 | 0.483 | 0.451 | 0.481 | 0.462 | 0.500 | 0.481 |

resolve the problem with the erroneous data when using TF-IDF strategy.

Furthermore, some of the statistical measures can be used as feature selection metrics when extracting data mining features from Linked Open Data. For example, considering the semantics of the resources, the IDF value can be computed upfront for all feature candidates, and can be used for selecting the most valuable features before the costly feature generation. Thus, intertwining propositionalization and feature selection strategies for Semantic Web knowledge graphs [253] will be an interesting line of future work.

In summary, this chapter has revealed some insights in a problem largely overlooked so far, i.e., choosing different propositionalization for mining Semantic Web knowledge graphs.

# Chapter 6

# Feature Selection in Hierarchical Feature Spaces

As introduced in the previous chapter, in machine learning and data mining, data is usually described as a vector of *features* or *attributes*, such as the age, income, and gender of a person. Based on this representation, predictive or descriptive models are built.

For many practical applications, the set of features can be very large, which leads to problems both with respect to the performance as well as the accuracy of learning algorithms. Thus, it may be useful to reduce the set of features in a preprocessing step, i.e., perform a *feature selection* [52, 182]. Usually, the goal is to compress the feature space as good as possible without a loss (or even with a gain) in the accuracy of the model learned on the data.

In some cases, external knowledge about attributes exist, in particular about their hierarchies. For example, a product may belong to different categories, which form a hierarchy (such as *Headphones* $<$ *Accessories* $<$ *Consumer Electronics*). Likewise, hyponym and hyperonym relations can be exploited when using bag-of-words features for text classification [133], or hierarchies defined by ontologies when generating features from Semantic Web knowledge graphs[225] in our case.

In this chapter, we introduce an approach that exploits hierarchies for feature selection in combination with standard metrics, such as *information gain* or *correlation*. With an evaluation on a number of synthetic and real world datasets, we show that using a combined approach works better than approaches not using the hierarchy, and also outperforms existing approaches for feature selection that exploit the hierarchy.

**The work presented in this chapter has been published before as: "Petar Ristoski, Heiko Paulheim:** ***A Comparison of Propositionalization Strategies for Creating Features from Linked Open Data.*** **Proceedings of the 1st Workshop on Linked Data for Knowledge Discovery co-located with European Conference on Machine Learning and Principles and Practice of Knowledge Discovery**

**in Databases (ECML PKDD 2014), Nancy, France, September 19th, 2014." [252].**

## 6.1 Problem Statement

We describe each instance as an $n$-dimensional binary feature vector $\langle v_1, v_2, ..., v_n \rangle$, with $v_i \in \{0, 1\}$ for all $1 \leq i \leq n$. We call $V = \{v_1, v_2, ..., v_n\}$ the *feature space*.

Furthermore, we denote a hierarchic relation between two features $v_i$ and $v_j$ as $v_i < v_j$, i.e., $v_i$ is more specific than $v_j$. For hierarchic features, the following implication holds:

$$v_i < v_j \rightarrow (v_i = 1 \rightarrow v_j = 1)\,, \tag{6.1}$$

i.e., if a feature $v_i$ is set, then $v_j$ is also set. Using the example of product categories, this means that a product belonging to a category also belongs to that product's super categories. Note that the implication is not symmetric, i.e., even if $v_i = 1 \rightarrow v_j = 1$ holds for two features $v_i$ and $v_j$, they do not necessarily have to be in a hierarchic relation. We furthermore assume transitivity of the hierarchy, i.e.,

$$v_i < v_j \wedge v_j < v_k \rightarrow v_i < v_k \tag{6.2}$$

The problem of *feature selection* can be defined as finding a projection of $V$ to $V'$, where $V' \subseteq V$. Ideally, $V'$ is much smaller than $V$.

Feature selection is usually regarded with respect to a certain problem, where a solution $S$ using a subset $V'$ of the features yields a certain performance $p(V')$, i.e., $p$ is a function

$$p : \mathcal{P}(V) \rightarrow [0, 1], \tag{6.3}$$

which is normalized to $[0, 1]$ without loss of generality. For example, for a classification problem, the accuracy achieved by a certain classifier on a feature subset can be used as the performance function $p$. Besides the quality, another interesting measure is the *feature space compression*, which we define as

$$c(V') := 1 - \frac{|V'|}{|V|} \tag{6.4}$$

Since there is a trade-off between the feature set and the performance, an overall target function is, e.g., the harmonic mean of $p$ and $c$.

For most problems, we expect the optimal features to be somewhere in the *middle* of the hierarchy, while the most general features are often too general for predictive models, and the most specific ones are too specific. The hierarchy level of the most valuable features depends on the task at hand. Fig. 6.1 shows a small part of the hierarchical feature space extracted for dataset Sports Tweets T (see section 6.3.1). If the task is to classify tweets into sports and non sports related, the optimal features are those in the upper rectangle, if the task is to classify them by different kinds of sports, then the features in the lower rectangle are more valuable.

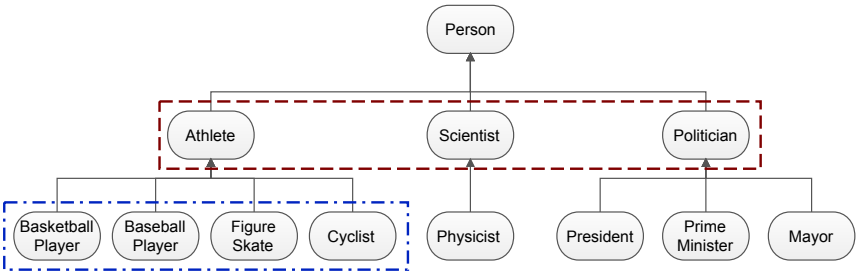



Figure 6.1: An example hierarchy of binary features

## 6.2 Approach

Following the implication shown in Eq. 6.1, we can assume that if two features subsume each other, they are usually highly correlated to each other and have similar relevance for building the model. Following the definition for "relevance" by Blum et al. [25], two features $v_i$ and $v_j$ have similar relevance if $1 - |R(v_i) - R(v_j)| \geq t,\ t \rightarrow [0, 1]$, where $t$ is a user specified threshold.

The core idea of our SHSEL approach is to identify features with similar relevance, and select the most valuable abstract features, i.e. features from as high as possible levels of the hierarchy, without losing predictive power. In our approach, to measure the similarity of relevance between two nodes, we use the standard correlation and information gain measure. The approach is implemented in two steps, i.e, initial selection and pruning. In the first step, we try to identify, and filter out the ranges of nodes with similar relevance in each branch of the hierarchy. In the second step we try to select only the most valuable features from the previously reduced set.

The initial selection algorithm is shown in Algorithm 1. The algorithm takes as input the feature hierarchy $H$, the initial feature set $F$, a relevance similarity threshold $t$, and the relevance similarity measure $s$ to be used by the algorithm. The relevance similarity threshold is used to decide whether two features would be similar enough, thus it controls how many nodes from different levels in the hierarchy will be merged. The algorithm starts with identifying the leaf nodes of the feature hierarchy. Then, starting from each leaf node $l$, it calculates the relevance similarity value between the current node and its direct ascendants $d$. The relevance similarity value is calculated using the selected relevance measure $s$. If the relevance similarity value is greater or equal to the similarity threshold $t$, then the node from the lower level of the hierarchy is removed from the feature space $F$. Also, the node is removed from the feature hierarchy $H$, and the paths in the hierarchy are updated accordingly. For the next iteration, the direct ascendants of the current node are added in the list $L$.

The algorithm for pruning is shown in Algorithm 2. The algorithm takes as input the feature hierarchy $H$ and the previously reduced feature set $F$. The algorithm starts with identifying all paths $P$ from all leaf nodes to the root node of

**Algorithm 1** Algorithm for initial hierarchy selection strategy.

**Data:** $H$: Feature hierarchy, $F$: Feature set, $t$: Importance similarity threshold, $s$:= Importance similarity measurement {"Information Gain", "Correlation"}
**Result:** $F$: Feature set

```
1  L := leaf nodes from hierarchy H
   foreach leaf l ∈ L do
2      D := direct ascendants of node l
       foreach node d ∈ D do
3          similarity = 0
           if s == "Information Gain" then
4              similarity = 1-ABS(IGweight(d)-IGweight(l))
5          else
6              similarity =Correlation(d,l)
7          end
8          if similarity ≥ threshold then
9              remove l from F
               remove l from H
               break
10         end
11     end
12     add direct ascendants of l to L
13 end
```

the hierarchy. Then, for each path $p$ it calculates the average information gain of all features on the path $p$. All features that have lower information gain than the average information gain on the path, are removed from the feature space $F$, and from the feature hierarchy $H$. In cases where a feature is located on more than one path, it is sufficient that the feature has greater information gain than the average information gain on at least one of the paths. This way, we prevent removing relevant features. Practically, the paths from the leafs to the root node, as well as the average information gain per path, can already be precomputed in the initial selection algorithm. The loop in the lines $3 - 6$ is only added for illustrating the algorithm.

Fig. 6.2a shows an example hierarchical feature set, with the information gain value of each feature. Applying the initial selection algorithm on that input hierarchical feature set, using information gain as a relevance similarity measurement, would reduce the feature set as shown in Fig. 6.2b. We can see that all feature pairs that have high relevance similarity value, are replaced with only one feature. However, the feature set still contains features that have a rather small relevance value. In Fig. 6.2c we can see that running the pruning algorithm, removes the

**Algorithm 2** Algorithm for pruning strategy.

**Data:** $H$: Feature hierarchy, $F$: Feature set
**Result:** $F$: Feature set
$L$ := leaf nodes from hierarchy $H$
$P := \emptyset$
**foreach** *leaf* $l \in L$ **do**
  $p$ = paths from $l$ to root of $H$
  add $p$ to $P$
**end**
**foreach** *path* $p \in P$ **do**
  $avg$ = Information gain average of path $p$
  **foreach** *node* $n \in$ *path* $p$ **do**
    **if** *IGweight(n)* $< avg$ **then**
      remove $n$ from $F$
      remove $n$ from $H$
    **end**
  **end**
**end**

unnecessary features.

For $n$ features and $m$ instances, iterating over the features, and computing the correlation or information gain with each feature's ancestor takes $O(am)$, given that a feature has an average of $a$ ancestors.[1] Thus, the overall computational complexity is $O(amn)$. It is, however, noteworthy that the selection of the features in both algorithms can be executed in parallel.

## 6.3 Evaluation

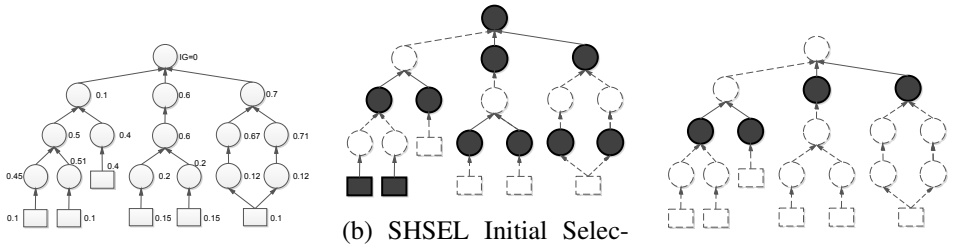
We perform an evaluation, both on real and on synthetic datasets, and compare different configurations of our approach to standard approaches for feature selection, as well as the approaches described in Section 13.2.

### 6.3.1 Datasets

In our evaluation, we used five real-world datasets and six synthetically generated datasets. The real-world datasets cover different domains, and are used for different classification tasks. Initially, the datasets contained only the instances with a given class label, which afterwards were extended with hierarchical features.

For generating the hierarchical features, we used the RapidMiner Linked Open Data extension [229], which is able to identify Linked Open Data resources inside

[1] $a$ is 1 in the absence of multiple inheritance, and close to 1 in most practical cases.




(a) Initial Feature Space (b) SHSEL Initial Selection (c) SHSEL Pruning

Figure 6.2: Illustration of the two steps of the proposed hierarchical selection strategy

the given datasets, and extract different types of features from any Linked Open Data source. In particular, we used DBpedia Spotlight [174], which annotates a text with concepts in DBpedia, a structured data version of Wikipedia [162]. From those, we can extract further features, such as the types of the concepts found in a text. For example, when the concept *Kobe Bryant* is found in a text, we can extract a hierarchy of types (such as *Basketball Player* $<$ *Athlete* $<$ *Person*), as well as a hierarchy of categories (such as *Shooting Guards* $<$ *Basketball* $<$ *Sports*). The generation of the features is independent from the class labels of the instances (i.e., the classification task), and it is completely unbiased towards any of the feature selection approaches.

The following datasets were used in the evaluation (see Table 6.1):

- *Sports Tweets T* dataset was used for existing Twitter topic classifier[2], where the classification task is to identify sports related tweets. The hierarchical features were generated by extracting all types of the discovered DBpedia concepts in each tweet.
- *Sports Tweets C* is the same dataset as the previous one, but using categories instead of types.
- The *Cities* dataset was compiled from the Mercer ranking list of the most and the least livable cities, as described in [219]. The classification task is to classify each city into high, medium, and low livability. The hierarchical features were generated by extracting the types for each city.
- The *NY Daily* dataset is a set of crawled news texts, which are augmented with sentiment scores[3]. Again, the hierarchical features were generated by extracting types.
- The *StumbleUpon* dataset is the training dataset used for the StumbleUpon Evergreen Classification Challenge[4]. To generate the hierarchical features, we used

[2]https://github.com/vinaykola/twitter-topic-classifier/blob/master/training.txt
[3]http://dws.informatik.uni-mannheim.de/en/research/identifying-disputed-topics-in-the-news
[4]https://www.kaggle.com/c/stumbleupon

Table 6.1: Evaluation Datasets.

| Name | Features | # Instances | Class Labels | # Features |
|---|---|---|---|---|
| *Sports Tweets T* | DBpedia Direct Types | 1,179 | positive(523); negative(656) | 4,082 |
| *Sports Tweets C* | DBpedia Categories | 1,179 | positive(523); negative(656) | 10,883 |
| *Cities* | DBpedia Direct Types | 212 | high(67); medium(106); low(39) | 727 |
| *NY Daily Headings* | DBpedia Direct Types | 1,016 | positive(580); negative(436) | 5,145 |
| *StumbleUpon* | DMOZ Categories | 3,020 | positive(1,370); negative(1,650) | 3,976 |

the Open Directory Project[5] to extract categories for each URL in the dataset.

To generate the synthetic datasets, we start with generating features in a flat hierarchy, i.e. all features are on the same level. The initial features were generated using a polynomial function, and then discretizing each attribute into a binary one. These features represent the *middle layer* of the hierarchy, which are then used to build the hierarchy upwards and downwards. The hierarchical feature implication (6.1) and the transitivity rule (6.2) hold for all generated features in the hierarchy. By merging the predecessors of two or more neighboring nodes from the middle layer, we are able to create more complex branches inside the hierarchy. We control the depth and the branching factor of the hierarchy with two parameters $D$ and $B$, respectively. Each of the datasets that we use for the evaluation contains 1000 instances, and contains 300 features in the middle layer. The datasets are shown in Table 6.2.

### 6.3.2 Experiment Setup

In order to demonstrate the effectiveness of our proposed feature selection in hierarchical feature space, we compare the proposed approach with the following methods:

- *CompleteFS*: the complete feature set, without any filtering.
- *SIG*: standard feature selection based on information gain value.
- *SC*: Standard feature selection based on feature correlation.
- *TSEL Lift*: tree selection approach proposed in [133], which selects the most representative features from each hierarchical branch based on the *lift* value.
- *TSEL IG*: this approach follows the same algorithm as *TSEL Lift*, but uses information gain instead of lift.
- *HillClimbing*: bottom-up hill-climbing approach proposed in [320].We use $k = 10$ for the kNN classifier used for scoring.
- *GreedyTopDown*: greedy based top-down approach described in [171], which tries to select the most valuable features from different levels of the hierarchy.

[5] http://www.dmoz.org/

Table 6.2: Synthetic Evaluation Datasets.

| Name | Feature Generation Strategy | # Instances | Classes | # Features |
|---|---|---|---|---|
| *S-D2-B2* | D=2; B=2 | 1,000 | positive(500); negative(500) | 1,201 |
| *S-D2-B5* | D=2; B=5 | 1,000 | positive(500); negative(500) | 1,021 |
| *S-D2-B10* | D=2; B=10 | 1,000 | positive(500); negative(500) | 961 |
| *S-D4-B2* | D=4; B=2 | 1,000 | positive(500); negative(500) | 2,101 |
| *S-D4-B5* | D=4; B=5 | 1,000 | positive(500); negative(500) | 1,741 |
| *S-D4-B10* | D=4; B=10 | 1,000 | positive(500); negative(500) | 1,621 |

- *initialSHSEL IG* and *initialSHSEL C*: our proposed initial selection approach shown with Algorithm 1, using information gain and correlation as relevance similarity measurement, respectively.

- *pruneSHSEL IG* and *pruneSHSEL C*: our proposed pruning selection approach shown with Algorithm 2, applied on previously reduced feature set, using *initialSHSEL IG* and *initialSHSEL C*, respectively.

For all algorithms involving a threshold (i.e., SIG, SC, and the variants of SHSEL), we use thresholds between 0 and 1 with a step width of 0.01.

For conducting the experiments, we used the RapidMiner machine learning platform and the RapidMiner development library. For SIG and SC, we used the built-in RapidMiner operators. The proposed approach for feature selection, as well as all other related approaches, were implemented in a separate operator as part of the RapidMiner Linked Open Data extension. All experiments were run using standard laptop computer with 8GB of RAM and Intel Core i7-3540M 3.0GHz CPU. The RapidMiner processes and datasets used for the evaluation can be found online[6].

### 6.3.3 Results

To evaluate how well the feature selection approaches perform, we use three classifiers for each approach on all datasets, i.e., Naïve Bayes, k-Nearest Neighbors (with $k = 3$), and Support Vector Machine. For the latter, we use Platt's sequential minimal optimization algorithm and a polynomial kernel function [236]. For each of the classifiers we were using the default parameters values in RapidMiner, and no further parameter tuning was undertaken. The classification results are calculated using stratified 10-fold cross validation, where the feature selection is performed separately for each cross-validation fold. For each approach, we report accuracy, feature space compression (6.4), and their harmonic mean.

[6] http://dws.informatik.uni-mannheim.de/en/research/feature-selection-in-hierarchical-feature-spaces

Table 6.3: Results on real world datasets

| | Sports Tweets T | | | Sports Tweets C | | | StumbleUpon | | | Cities | | | NY Daily Headings | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NB | KNN | SVM | NB | KNN | SVM | NB | KNN | SVM | NB | KNN | SVM | NB | KNN | SVM |
| *Classification Accuracy* | | | | | | | | | | | | | | | |
| CompleteFS | .655 | .759 | .797 | .943 | .920 | .946 | .582 | .699 | .730 | .625 | .562 | .684 | .534 | .586 | .577 |
| initialSHSEL IG | **.836** | .768 | **.824** | **.974** | .768 | **.953** | .661 | .709 | .733 | .671 | .609 | .674 | **.688** | .629 | .635 |
| initialSHSEL C | .819 | .765 | .811 | .946 | **.937** | **.953** | .689 | **.723** | .732 | .640 | .671 | .683 | .547 | .580 | .596 |
| pruneSHSEL IG | .791 | **.793** | .773 | .909 | .909 | .946 | **.717** | .695 | **.737** | **.687** | .669 | **.689** | **.688** | **.659** | **.671** |
| pruneSHSEL C | .786 | .791 | .772 | .946 | .918 | .935 | .711 | .707 | .732 | .656 | **.687** | .646 | .665 | **.659** | .661 |
| SIG | .819 | .788 | .814 | .966 | .936 | .940 | .681 | .707 | .729 | .656 | .640 | .671 | .675 | .652 | .668 |
| SC | .816 | .765 | .813 | .937 | .918 | .932 | .587 | .711 | .726 | .625 | .656 | .677 | .534 | .583 | .606 |
| TSEL Lift | .641 | .740 | .787 | .836 | .855 | .893 | .570 | .613 | .690 | 0 | 0 | 0 | .498 | .544 | .565 |
| TSEL IG | .632 | .734 | .782 | .923 | .909 | .935 | .579 | .661 | .724 | .640 | .580 | .580 | .521 | .560 | .610 |
| HillClimbing | .528 | .647 | .742 | .823 | .836 | .876 | .548 | .653 | .683 | .622 | .562 | .551 | .573 | .583 | .530 |
| GreedyTopDown | .658 | .788 | .800 | .943 | .929 | .944 | .582 | .698 | .727 | .625 | .562 | .679 | .534 | .570 | .595 |
| *Feature Space Compression* | | | | | | | | | | | | | | | |
| initialSHSEL IG | .456 | .207 | .222 | .318 | .708 | .288 | .672 | .843 | .642 | .781 | **.902** | .779 | .858 | .322 | .631 |
| initialSHSEL C | .231 | .173 | .290 | .321 | .264 | .228 | **.993** | .445 | .644 | .184 | .121 | .116 | .285 | .572 | .790 |
| pruneSHSEL IG | **.985** | **.986** | **.969** | .895 | **.907** | **.916** | .976 | .957 | **.975** | **.823** | .466 | .452 | .912 | .817 | .817 |
| pruneSHSEL C | .971 | .965 | .965 | **.897** | .857 | .861 | .966 | **.968** | .959 | .305 | .265 | .308 | .519 | .586 | .566 |
| SIG | .360 | .741 | .038 | .380 | .847 | .574 | .940 | .615 | .604 | .774 | .775 | 04 | .240 | .289 | .565 |
| SC | .667 | .712 | .635 | .887 | .710 | .792 | .585 | .821 | .712 | .631 | .704 | .598 | .632 | .927 | .620 |
| TSEL Lift | .247 | .247 | .247 | .511 | .511 | .511 | .412 | .412 | .412 | 0 | 0 | 0 | **.956** | **.956** | **.956** |
| TSEL IG | .920 | .920 | .920 | .522 | .522 | .522 | .471 | .471 | .471 | .126 | .126 | .126 | .926 | .926 | .926 |
| HillClimbing | .770 | .770 | .770 | .748 | .748 | .748 | .756 | .756 | .756 | .817 | .817 | **.817** | .713 | .713 | .713 |
| GreedyTopDown | .136 | .136 | .136 | .030 | 0.030 | .030 | .285 | .285 | .285 | .048 | .048 | .048 | .135 | .135 | .135 |
| *Harmonic Mean of Classification Accuracy and Feature Space Compression* | | | | | | | | | | | | | | | |
| initialSHSEL IG | .590 | .326 | .350 | .480 | .737 | .442 | .666 | .770 | .684 | .722 | **.727** | **.723** | **.764** | .426 | .633 |
| initialSHSEL C | .360 | .282 | .427 | .479 | .412 | .368 | .814 | .551 | .686 | .286 | .205 | .199 | .375 | .576 | .679 |
| pruneSHSEL IG | **.877** | **.879** | **.860** | .902 | **.908** | **.931** | **.827** | .805 | **.840** | **.749** | .549 | .546 | .784 | **.729** | **.737** |
| pruneSHSEL C | .869 | .869 | .858 | **.921** | .886 | .896 | .820 | **.817** | .830 | .416 | .383 | .417 | .583 | .620 | .610 |
| SIG | .500 | .764 | .073 | .545 | .889 | .713 | .789 | .658 | .660 | .710 | .701 | 08 | .354 | .401 | .612 |
| SC | .734 | .738 | .713 | .911 | .801 | .856 | .586 | .762 | .719 | .628 | .679 | .635 | .579 | .716 | .613 |
| TSEL Lift | .356 | .370 | .376 | .634 | .640 | .650 | .479 | .493 | .516 | 0 | 0 | 0 | .655 | .693 | .711 |
| TSEL IG | .749 | .817 | .846 | .667 | .663 | .670 | .520 | .550 | .571 | .211 | .207 | .207 | .667 | .698 | .735 |
| HillClimbing | .626 | .703 | .756 | .784 | .790 | .807 | .636 | .701 | .718 | .706 | .666 | .658 | .635 | .641 | .608 |
| GreedyTopDown | .225 | .232 | .232 | 0.058 | .058 | .058 | .383 | .405 | .409 | .089 | .088 | .089 | .216 | .219 | .221 |

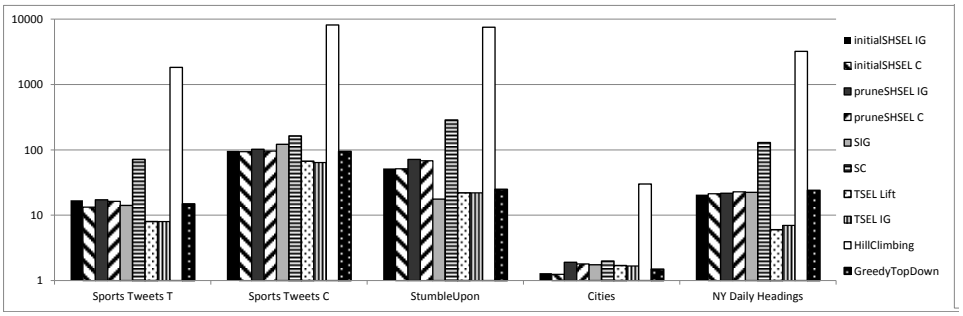



Figure 6.3: Runtime (seconds) - Real World Datasets

### Results on Real World Datasets

Table 6.3 shows the results of all approaches. Because of the space constrains, for the *SIG* and *SC* approaches, as well as for our proposed approaches, we show only the best achieved results. The best results for each classification model are marked in bold. As we can observe from the table, our proposed approach outperforms all other approaches in all five datasets for both classifiers in terms of accuracy. Furthermore, we can conclude that our proposed approach delivers the best feature space compression for four out of five datasets. When looking at the harmonic mean, our approach also outperforms all other approaches, most often with a large gap. From the results for the harmonic mean we can conclude that the *pruneSHSEL IG* approach, in most of the cases, delivers the best results

Additionally, we report the runtime of all approaches on different datasets in Fig. 6.3. The runtime of our approaches is comparable to the standard feature selection approach, *SIG*, runtime. The *HillClimbing* approach has the longest runtime due to the repetitive calculation of the kNN for each instance. Also, the standard feature selection approach *SC* shows a long runtime, which is due to the computation of correlation between all pairs of features in the feature set.

### Results on Synthetic Datasets

Table 6.4 shows the results for the different synthetic datasets. Our approaches achieve the best results, or same results as the standard feature selection approach *SIG*. The results for the feature space compression are rather mixed, while again, our approach outperforms all other approaches in terms of the harmonic mean of accuracy and feature space compression. The runtimes for the synthetic datasets, which we omit here, show the same characteristics as for the real-world datasets.

Overall, *pruneSHSEL IG* delivers the best results on average, with an *importance similarity threshold* $t$ in the interval $[0.99; 0.9999]$. When using correlation, the results show that $t$ should be chosen greater than $0.6$. However, the selection of the approach and the parameters' values highly depends on the given dataset, the given data mining task, and the data mining algorithm to be used.

Table 6.4: Results on synthetic datasets

| | S_D2_B2 | | | S_D2_B5 | | | S_D2_B10 | | | S_D4_B2 | | | S_D4_B5 | | | S_D4_B10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NB | KNN | SVM | NB | KNN | SVM | NB | KNN | SVM | NB | KNN | SVM | NB | KNN | SVM | NB | KNN | SVM |
| *Classification Accuracy* | | | | | | | | | | | | | | | | | | |
| CompleteFS | .565 | .500 | .500 | .700 | .433 | .530 | .600 | .466 | .610 | .666 | .600 | .560 | .566 | .566 | .600 | .533 | .466 | .630 |
| initialSHSEL IG | **1.0** | .833 | .880 | **1.0** | .766 | .850 | **1.0** | **.866** | .890 | **1.0** | **.866** | .880 | **1.0** | **.936** | .870 | .956 | .733 | .910 |
| initialSHSEL C | .666 | .633 | .833 | .700 | .633 | .740 | .666 | .633 | .780 | .766 | .666 | .740 | .600 | .633 | .730 | .633 | .533 | .860 |
| pruneSHSEL IG | **1.0** | **.933** | **.920** | **1.0** | **.800** | **.910** | **1.0** | .833 | **.960** | **1.0** | **.866** | **.960** | **1.0** | .866 | **.980** | **1.0** | **.800** | **.986** |
| pruneSHSEL C | .866 | .666 | .910 | .866 | .700 | .900 | .800 | .766 | .900 | .800 | .766 | .910 | .933 | .833 | .880 | .933 | .666 | .930 |
| SIG | .960 | .900 | .830 | **1.0** | **.800** | .900 | .930 | .766 | .933 | **1.0** | .833 | .933 | **1.0** | .900 | .966 | **1.0** | .733 | .966 |
| SC | .700 | .700 | .733 | .700 | .666 | .733 | .730 | .600 | .700 | .733 | .666 | .700 | .700 | .666 | .766 | .700 | .700 | .733 |
| TSEL Lift | .553 | .500 | .540 | .633 | .666 | .630 | .400 | .500 | .540 | .566 | .533 | .540 | .500 | .566 | .510 | .466 | .533 | .480 |
| TSEL IG | .866 | .533 | .810 | .666 | .566 | .700 | .733 | .500 | .770 | .766 | .666 | .720 | .533 | .600 | .700 | .500 | .566 | .710 |
| HillClimbing | .652 | .633 | .630 | .633 | .636 | .580 | .633 | .566 | .640 | .676 | .566 | .620 | .676 | .534 | .586 | .689 | .523 | .590 |
| GreedyTopDown | .666 | .600 | .800 | .703 | .633 | .780 | .633 | .466 | .830 | .703 | .566 | .820 | .752 | .700 | .850 | .833 | .500 | .830 |
| *Feature Space Compression* | | | | | | | | | | | | | | | | | | |
| initialSHSEL IG | .846 | .572 | .864 | **.948** | .907 | .880 | .861 | .810 | **.886** | .914 | .789 | .740 | .929 | .868 | .746 | .912 | .750 | .918 |
| initialSHSEL C | .267 | .557 | .875 | .104 | .888 | **.938** | .279 | **.956** | .656 | .441 | .893 | .890 | .831 | .742 | .786 | .627 | .805 | .805 |
| pruneSHSEL IG | **.930** | .911 | .796 | .925 | **.933** | .824 | **.899** | .944 | .850 | **.956** | .877 | .877 | .955 | **.969** | .863 | .956 | .873 | .791 |
| pruneSHSEL C | .697 | .896 | .800 | .639 | .636 | .667 | .781 | .696 | .823 | .795 | .776 | .849 | .692 | .726 | .742 | .731 | .826 | .750 |
| SIG | .922 | **.922** | .861 | .842 | .842 | .753 | .865 | .930 | .865 | .886 | .595 | .708 | .891 | .719 | .525 | .900 | .704 | .704 |
| SC | .717 | .909 | **.880** | .693 | .900 | .159 | .750 | .692 | .869 | .379 | .493 | .769 | .628 | .736 | .742 | .667 | .727 | .289 |
| TSEL Lift | .750 | .750 | .750 | .706 | .706 | .706 | .687 | .687 | .687 | .857 | .857 | .857 | .827 | .827 | .827 | .814 | .814 | .814 |
| TSEL IG | .836 | .836 | .836 | .866 | .866 | .866 | .856 | .856 | .856 | .926 | **.926** | **.926** | **.965** | .965 | **.965** | **.970** | **.970** | **.970** |
| HillClimbing | .770 | .770 | .770 | .751 | .751 | .751 | .805 | .805 | .805 | .792 | .792 | .792 | .776 | .776 | .776 | .795 | .795 | .795 |
| GreedyTopDown | .399 | .399 | .399 | .370 | .370 | .370 | .356 | .356 | .356 | .470 | .470 | .470 | .404 | .404 | .404 | .438 | .438 | .438 |
| *Harmonic Mean of Classification Accuracy and Feature Space Compression* | | | | | | | | | | | | | | | | | | |
| initialSHSEL IG | .917 | .679 | .872 | .973 | .831 | .865 | .925 | .837 | .888 | .955 | .826 | .804 | .963 | .901 | .803 | .933 | .741 | **.914** |
| initialSHSEL C | .381 | .592 | .853 | .182 | .739 | .827 | .394 | .762 | .713 | .560 | .763 | .808 | .697 | .683 | .757 | .630 | .641 | .832 |
| pruneSHSEL IG | **.964** | **.922** | **.854** | **.961** | **.861** | **.865** | **.946** | **.885** | **.901** | **.977** | **.871** | **.916** | **.977** | **.915** | **.918** | **.977** | **.835** | .878 |
| pruneSHSEL C | .773 | .764 | .851 | .736 | .666 | .766 | .790 | .729 | .859 | .797 | .771 | .878 | .795 | .776 | .805 | .819 | .737 | .830 |
| SIG | .940 | .911 | .845 | .914 | .820 | .820 | .896 | .840 | .898 | .940 | .694 | .805 | .942 | .799 | .680 | .947 | .718 | .815 |
| SC | .708 | .791 | .800 | .696 | .766 | .262 | .740 | .642 | .775 | .500 | .567 | .733 | .662 | .700 | .754 | .683 | .713 | .415 |
| TSEL Lift | .636 | .600 | .628 | .667 | .685 | .665 | .505 | .579 | .605 | .682 | .657 | .662 | .623 | .672 | .631 | .593 | .644 | .604 |
| TSEL IG | .851 | .651 | .822 | .753 | .685 | .774 | .790 | .631 | .810 | .839 | .775 | .810 | .687 | .740 | .811 | .660 | .715 | .820 |
| HillClimbing | .706 | .695 | .693 | .687 | .689 | .654 | .709 | .665 | .713 | .730 | .660 | .695 | .723 | .633 | .668 | .738 | .631 | .677 |
| GreedyTopDown | .499 | .479 | .533 | .485 | .467 | .502 | .456 | .404 | .499 | .564 | .514 | .598 | .526 | .513 | .548 | .574 | .467 | .573 |

## 6.4 Conclusion and Outlook

In this chapter, we have proposed a feature selection method exploiting hierarchic relations between features. It runs in two steps: it first removes redundant features along the hierarchy's paths, and then prunes the remaining set based on the features' predictive power. Our evaluation has shown that the approach outperforms standard feature selection techniques as well as with recent approaches which use hierarchies.

So far, we have only considered classification problems. A generalizing of the pruning step to tasks other than classification would be an interesting extension. While a variant for regression tasks seems to be rather straight forward, other problems, like association rule mining, clustering, or outlier detection, would probably require entirely different pruning strategies.

Furthermore, we have only regarded simple hierarchies so far. When features are organized in a complex ontology, there are other relations as well, which may be exploited for feature selection. Generalizing the approach to *arbitrary* relations between features is also a relevant direction of future work.

# Chapter 7

# Mining the Web of Linked Data with RapidMiner

As shown in chapter 2, the Web of Linked Data contains a collection of machine processable, interlinked datasets from various domains, ranging from general cross-domain knowledge sources to government, library and media data, which today comprises roughly a thousand datasets [21, 267]. While many domain-specific applications use Linked Open Data, general-purpose applications rarely go beyond displaying the mere data, and provide little means of deriving additional knowledge from the data.

At the same time, sophisticated data mining platforms exist, which support the user with finding patterns in data, providing meaningful visualizations, etc. What is missing is a *bridge* between the vast amount of data on the one hand, and intelligent data analysis tools on the other hand. Given a data analysis problem, a data analyst should be able to automatically find suitable data from different relevant data sources, which will then be combined and cleansed, and served to the user for further analysis. This data collection, preparation, and fusion process is an essential part of the data analysis workflow [85], however, it is also one of the most time consuming parts, constituting roughly half of the costs in data analytics projects [35]. Furthermore, since the step is time consuming, a data analyst most often makes a heuristic selection of data sources based on his a priori assumptions, and hence is subject to the selection bias. Despite these issues, automation at that stage of the data processing step is still rarely achieved.

In this chapter, we discuss how the Web of Linked Data can be mined using the full functionality of the state of the art data mining environment *RapidMiner*[1] [122]. We introduce an extension to RapidMiner, which allows for bridging the gap between the Web of Data and data mining, and which can be used for carrying out sophisticated analysis tasks on and with Linked Open Data. The extension provides means to automatically connect local data to background knowledge from Linked Open Data, or load data from the desired Linked Open Data source into the

[1] `http://www.rapidminer.com/`

RapidMiner platform, which itself provides more than 400 operators for analyzing data, including classification, clustering, and association analysis.

RapidMiner is a programming-free data analysis platform, which allows the user to design data analysis processes in a plug-and-play fashion by wiring *operators*. Furthermore, functionality can be added to RapidMiner by developing extensions, which are made available on the *RapidMiner Marketplace*[2]. The *RapidMiner Linked Open Data extension* adds operators for loading data from datasets within Linked Open Data, as well as autonomously following RDF links to other datasets and gathering additional data from there. Furthermore, the extension supports schema matching for data gathered from different datasets.

As the operators from that extension can be combined with all RapidMiner built-in operators, as well as those from other extensions (e.g., for time series analysis), complex data analysis processes on Linked Open Data can be built. Such processes can automatically combine and integrate data from different datasets and support the user in making sense of the integrated data.

**The work presented in this chapter has been published before as: "Petar Ristoski, Christian Bizer, Heiko Paulheim: *Mining the web of linked data with rapidminer.* Web Semantics: Science, Services and Agents on the World Wide Web, Vol 35, pages 142–151, 2015." [246].**

## 7.1 Description

RapidMiner is a data mining platform, in which data mining and analysis processes are designed from elementary building blocks, so called *operators*. Each operator performs a specific action on data, e.g., loading and storing data, transforming data, or inferring a model on data. The user can compose a process from operators by placing them on a canvas and wiring their input and output ports, as shown in Fig. 7.2.

The *RapidMiner Linked Open Data* extension adds a set of operators to RapidMiner, which can be used in data mining processes and combined with RapidMiner built-in operators, as well as other operators. The operators in the extension fall into different categories: data import, data linking, feature generation, schema matching, and feature subset selection.

### 7.1.1 Data Import

RapidMiner itself provides import operators for different data formats (e.g., Excel, CSV, XML). The Linked Open Data extension adds two import operators:

- A *SPARQL Importer* lets the user specify a SPARQL endpoint or a local RDF model, and a SPARQL query, and loads the query results table into RapidMiner.

[2] `https://marketplace.rapidminer.com/`

For local RDF models, SPARQL queries can also be executed with RDFS and different flavors of OWL inference [124].

- A *Data Cube Importer* can be used for datasets published using the RDF Data Cube vocabulary[3]. Following the Linked Data Cube Explorer (LDCX) prototype described in [139], the importer provides a wizard which lets the user select the dimensions to use, and creates a pivot table with the selected data.

### 7.1.2 Data Linking

In order to combine a local, potentially non-RDF dataset (e.g., data in a CSV file or a database) with data from the LOD cloud, links from the local dataset to remote LOD cloud datasets have to be established first. For that purpose, different linking operators are implemented in the extension:

- The *Pattern-based linker* creates URIs based on a string pattern. If the pattern a dataset uses for constructing its URIs is known, this is the fastest and most accurate way to construct URIs. For example, the *RDF Book Mashup* [20] dataset uses a URI pattern for books which is based on the ISBN.[4]

- The *Label-based linker* searches for resources whose label is similar to an attribute in the local dataset, e.g., the product name. It can only be used on datasets providing a SPARQL interface and is slower than the pattern-based linker, but can also be applied if the link patterns are not known, or cannot be constructed automatically.

- The *Lookup linker* uses a specific search interface[5] for the *DBpedia* dataset [162]. It also finds resources by alternative names (e.g., *NYC* or *NY City* for *New York City*). For DBpedia, it usually provides the best accuracy.

- For processing text, a linker using *DBpedia Spotlight*[6] [174] has also been included, which identifies multiple DBpedia entities in a textual attribute.

- The *SameAs linker* can be used to follow links from one dataset to another. Since many datasets link to DBpedia, a typical setup to link to an arbitrary LOD dataset is a two-step approach: the Lookup linker first establishes links to DBpedia at high accuracy. Then, `owl:sameAs` links between DBpedia and the target dataset are exploited to set the links to the latter.

[3] `http://www.w3.org/TR/vocab-data-cube/`

[4] In cases where additional processing is required, such as removing dashes in an ISBN, the operator may be combined with the built-in *Generate Attributes* operator, which can perform such operations.

[5] `http://lookup.dbpedia.org/`

[6] `http://spotlight.dbpedia.org/`

### 7.1.3 Feature Generation

For creating new data mining features from Linked Open Data sources, different strategies are implemented in the extension's operators (some already described in Chapter 5):

- The *Direct Types* generator extracts all types (i.e., objects of `rdf:type`) for a linked resource. For datasets such as YAGO[7], those types are often very informative, for example, products may have concise types such as *Smartphone* or *AndroidDevice*.

- The *Datatype Properties* generator extracts all datatype properties, i.e., numerical and date information (such as the price and release date of products).

- The *Relations* generator creates a binary or a numeric attribute for each property that connects a resource to other resource. For example, if a dataset contains awards won by products, an *award* attribute would be generated, either as a binary feature stating whether the product won an award or not, or a numerical one stating the number of awards.

- The *Qualified Relations* generator also generates binary or numeric attributes for properties, but takes the type of the related resource into account. For example, for a product linked to a manufacturer of type *GermanCompany*, a feature stating whether the product has been manufactured by a German company or not would be created.

- The *Specific Relations* generator creates features for a user-specified relation, such as Wikipedia categories included in DBpedia. It also allows for following property chains spanned by a user-defined property, e.g., following the `skos:broader` relation for Wikipedia categories in DBpedia in order to extract features for super-categories as well.

- In addition to those automatic generators, it is also possible to control the attribute generation process in a more fine-grained manner by issuing specific SPARQL queries using the *Custom SPARQL* generator.

*Kernel functions* are distance functions that hold between two data instances in some arbitrary space. They are commonly used in Support Vector Machines (SVMs), in which they allow training a classifier without explicitly transforming the data into a feature vector representation [48]. However, kernel functions *can* be used to transform, e.g., RDF data into a propositional feature vector. The Linked Open Data Extension integrates such operators for two types of kernels from the *Data2Semantics Mustard library*[8]:

[7] `http://yago-knowledge.org`
[8] `https://github.com/Data2Semantics/mustard`

- *RDF Walk Count Kernel* counts the different walks in the subgraphs (up to the provided graph depth) around the instances nodes. The maximum length of the walks can be specified as a parameter. For this kernel, there are four different variants:
  - *Fast*: This is a fast approximation of counting all the walks in the subgraph, which is done with the Full setting.
  - *Root*: Only considers walks that start with the instance node (i.e. the root) [58].
  - *Tree*: Counts all the walks in the subtree that is rooted in the instance node. This is faster than the Full subgraph version, since a tree does not contain cycles.
  - *Full*: Counts all the walks in the subgraph.
- *RDF WL Sub Tree Kernel* counts the different full subtrees in the subgraphs (up to the provided graph depth) around the instances nodes, using the Weisfeiler-Lehman algorithm [277]. The maximum size of the subtrees is controlled by the number of iterations, which can be specified parameter. Like for the RDF Walk Count Kernel, there are four different variants of this kernel:
  - *Fast*: This is a fast approximation of counting all the subtrees in the subgraph, which is done with the Full setting [57].
  - *Root*: Only considers subtrees that start with the instance node (i.e. the root).
  - *Tree*: Counts all the subtrees in the subtree that is rooted in the instance node. This is faster than the Full subgraph version, since a tree does not contain cycles.
  - *Full*: Counts all the subtrees in the subgraph.

  All of those feature generation operators can work in three different modes:

  1. using a predefined SPARQL endpoint,
  2. dereferencing URIs and processing the RDF returned from an HTTP request, or
  3. using a local RDF model.

While the SPARQL-based variant is usually faster, the dereferencing URIs variant is more versatile, as it can also work be applied to datasets not offering a SPARQL endpoint.

Furthermore, all mentioned generators are able to retrieve the hierarchical relations between the features, e.g., the *direct types* generator examines the `rdfs:subClassOf` property. Such hierarchies may be exploited for feature selection (see below). The extension also offers a graphical visualization of such hierarchies that might be useful for a better understanding of the data. For example, an excerpt of the retrieved

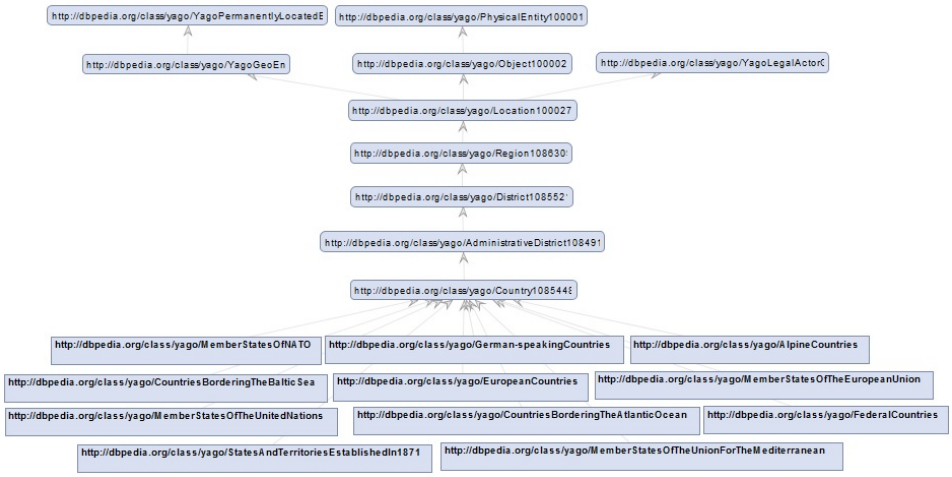



Figure 7.1: Hierarchical relations (excerpt) between the features retrieved from the YAGO ontology in DBpedia for instances of type *Country*, visualized in RapidMiner.

hierarchy from the YAGO ontology in DBpedia for instances of type *Country* is depicted in Figure 7.1.

When generating data mining features from graph based data, different *propositionalization strategies* can be applied. For example, the standard binary or numeric (i.e., counting) representation can be used, or more sophisticated representation strategies that use some graph characteristics might be introduced. The strategy of creating features has an influence on the data mining result. For example, proximity-based algorithms like k-NN will behave differently depending on the strategy used to create numerical features, as the strategy has a direct influence on most distance functions.

So far, the extension supports four strategies for generating a feature for properties connected with a resource:

- Creating a *binary feature* for each relation.
- Creating a *count feature* for each relation, specifying the number of resources connected by this relation.
- Creating a *relative count feature* for each relation, specifying the fraction of resources connected by this relation. For a resource that has total number of $P$ relations, the relative count value for a relation $p$ is defined as $\frac{n_p}{P}$, where $n_p$ is the number of relations of type $p$.
- Creating a *TF-IDF feature* for each relation, whose value is $\frac{n_p}{P} \cdot log \frac{N}{|\{r|\exists r' : p(r,r')\}|}$, where $N$ is the overall number of resources, and $|\{r|\exists r' : p(r,r')\}|$ denotes the number of resources for which the relation $p$ exists.

In [252], we have shown that the propositionalization strategy has an influence on the result quality of a data mining process (see also section 7.3.2).

### 7.1.4 Feature Subset Selection

All standard methods for feature subset selection can be used in conjunction with the RapidMiner Linked Open Data extension, as well as specialized operators from the *Feature Subset Selection extension*[9]. Furthermore, the Linked Open Data extension provides the *Simple Hierarchy Filter*, which exploits the schema information of a Linked Open Data source, and often achieves a better compression of the feature set than standard, non-hierarchical operators, without losing valuable features. The operator implements different hierarchical feature selection strategies [133, 253, 320]. As shown in [253] and Chapter 5, those algorithms often lead to a better compression of the feature space and a reduction of overfitting by avoiding the selection of too specific features (see also section 7.3.3).

### 7.1.5 Exploring Links

The feature generation algorithms above so far use only one input URI, and obtain features from that URI. This means that they are usually restricted to *one* dataset. For making use of the entire LOD cloud, the extension provides the *Link Explorer* meta operator, which follows links of a given type (by default: `owl:sameAs`) to a specified depth, and applies a set of operators to each resource discovered by that (see Fig. 7.2). A typical configuration is to use the link explorer in combination with the datatype properties generator, which results in following links from one starting point, and collect all the datatype properties for all linked resources.

Since the datasets that are used by that meta operator are not known a priori, and there is no reliable way of discovering a SPARQL endpoint given a resource URI [226], the link explorer only works by derefencing URIs, but not by means of SPARQL queries.

### 7.1.6 Data Integration

When combining data from different LOD sources, those usually use different schemas. For example, the population of a country can be contained in different datasets, using a different datatype property to denote the information. In order to use that data more effectively, such attributes can be merged into one by applying schema matching. The extension provides the *PARIS LOD Matcher*, which is an adaptation of the PARIS framework [290], and is able to perform alignment of instances, relations and classes. The matching operator outputs all discovered correspondences. Those can then be passed to the *Data Fusion* operator, which offers various conflict resolution strategies [22], e.g., majority voting, average, median, etc.

[9] `http://sourceforge.net/projects/rm-featselext/`

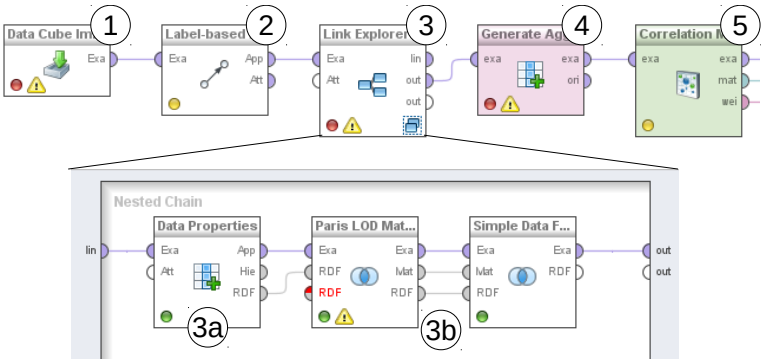



Figure 7.2: Overview of the process used in the running example, including the nested subprocess in the link explorer operator

## 7.2 Example Use Case

In our example use case, we use an RDF data cube with World Bank economic indicators data[10] as a starting point. The data cube contains time-indexed data for more than 1,000 indicators in over 200 countries. As shown in Figure 7.2, the process starts with importing data from that data cube (1). To that end, a wizard is used, which lets the user select the indicator(s) of interest. The complete data cube import wizard is shown in Figure 7.3. In our example, in the first step we select the indicator "Scientific and technical journal articles", which reports the number of such articles per country and year. In the second step, we select the dimensions and the measures.

As a row dimension, we select the countries, and as column dimensions, we select the time-indexed values for the number of scientific and technical journal articles. In the final step of the import wizard, we are able to select the values for the previously selected dimensions. After completing the wizard, the resulting table is generated. The indicator is present for 165 countries, so our resulting data table contains 165 rows, with columns per year, depicting the target value from 1960 to 2011. We are interested in understanding which factors drive a large *increase* in that indicator.

In the next step, we set links of the data imported from the RDF cube to other datasets (2). In our example, we use the label-based linker to find countries in DBpedia which have the same name as the country in the imported slice of the data cube.

The subsequent step is identifying more relevant datasets by following RDF links, and getting the data from there. This is carried out by the link explorer operator (3). Starting from DBpedia, we follow all `owl:sameAs` links to a depth of 2. Inside the link explorer, we collect datatype properties (3a) and also perform

[10] `http://worldbank.270a.info`

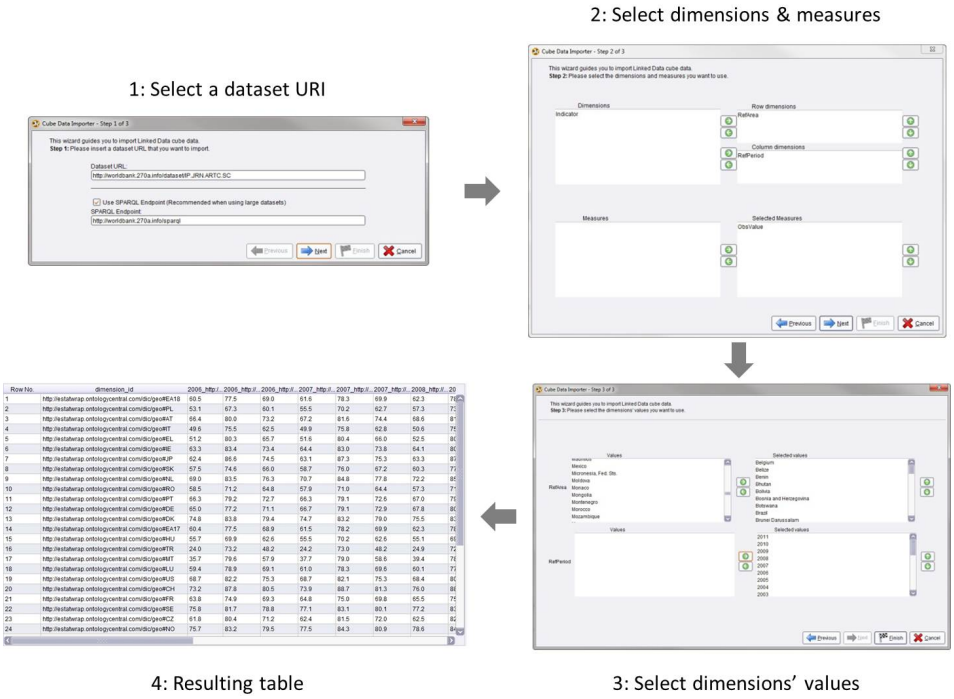



Figure 7.3: Data cube import wizard

the matching (3b). We end up with data from ten different datasets, i.e., DBpedia[11], LinkedGeoData[12], Eurostat[13], Geonames[14], WHO's Global Health Observatory[15], Linked Energy Data[16], OpenCyc[17], World Factbook[18], and YAGO[19].

The initial set of datatype properties extracted has $1,329$ attributes, which, after processing by schema matching and conflict resolution, are fused to $1,090$ attributes. For example, we find six different sources stating the population of countries by following RDF links between datasets, five of which are merged into one single attribute. A detailed evaluation of the feature consolidation process is given in Section 7.3.4

Once all the attributes have been extracted and matched, the actual analysis starts. First, the *Generate Aggregate* operator (4), which is a RapidMiner built-in operator, computes the *increase* in scientific publications from the individual per-

[11] http://dbpedia.org
[12] http://linkedgeodata.org
[13] http://eurostat.linked-statistics.org/ and http://wifo5-03.informatik.uni-mannheim.de/eurostat/
[14] http://sws.geonames.org/
[15] http://gho.aksw.org/
[16] http://en.openei.org/lod/
[17] http://sw.opencyc.org/
[18] http://wifo5-03.informatik.uni-mannheim.de/factbook/
[19] http://yago-knowledge.org

year values. Then, a correlation matrix is computed (5), again with a RapidMiner built-in operator, to find interesting factors that explain an increase in scientific publications.

From all the additional attributes we found by following links to different datasets in the Web of Data, we are now able to identify various attributes that explain a strong increase in scientific publications:

- The fragile state index (FSI) and the Human Development Index (HDI) are aggregate measures comprised of different social, political and health indicators, and both are good indicators for the growth of scientific publications.
- The GDP per capita is also strongly correlated with the increase in scientific publications. This hints at wealthier countries being able to invest more federal money into science funding.
- For European countries, the number of EU seats shows a significant correlation with the increase in scientific publications. As larger countries have more seats (e.g,. Germany, France, UK), this may hint at an increasing fraction of EU funding for science going being attributed to those countries [34].
- Additionally, many climate indicators show a strong correlation with the increase in scientific publications: precipitation has a negative correlation with the increase, while hours of sun and temperature averages are positively correlated. This can be explained by an unequal distribution of wealth across different climate zones [262], with the wealthier nations often located in more moderate climate zones.[20]

So far, these results have concentrated on one specific world bank indicator, while, as stated above, there are more than a thousand. We have conducted similar experiments with other indicators as well, revealing different findings. For example, we looked into the savings of energy consumption over the last years. Here, we can observe, e.g., a correlation with the GDP, showing that wealthier countries can afford putting more efforts into saving energy, and also have the economic means to replace old, energy inefficient infrastructure with new, more energy efficient facilities[21]

In summary, the experiment shows that

- we can enrich a dataset at hand with external knowledge from the LOD cloud,
- we can follow RDF links between datasets, and, by that, gather and combine data from different sources to solve a task at hand, and
- we can use analysis methods that identify relevant answers to a question.

[20] A more tongue-in-cheek interpretation may be that if the weather is bad, scientists spend more time in the lab writing journal articles.

[21] See., e.g., `http://www.forbes.com/2008/07/03/energy-efficiency-japan-biz-energy_cx_jz_0707efficiency_countries.html`

## 7.3 Evaluation

Many of the algorithms implemented by the RapidMiner LOD extension have been evaluated in different settings. In this section, we point out the key evaluations for the most important features of the extension, and introduce additional evaluations of specific aspects of the extension.

### 7.3.1 Feature Generation

The feature generation strategies have been evaluated in a prior publication. In [229] we used the *Auto MPG* dataset[22], a dataset that captures different characteristics of cars (such as cyclinders, transmission, horsepower), and the target is to predict the fuel consumption in Miles per Gallon (MPG). The original dataset contains 398 cars, each having a name, seven data attributes, and the MPG target attribute. The goal is to predict the fuel consumption from the characteristics of the car, i.e., a regression model has to be learned. Using our extension, we added different new attributes from the DBpedia dataset, i.e., direct types and categories.

For the prediction of the MPG attribute, we showed that when using the M5Rules algorithm [123], the relative error of the prediction is only half as large as the error on the original, non-enriched data.

The new attributes also provide insights that are not possible from the original dataset alone. For example, UK cars have a lower consumption than others (while the *origin* attribute contained in the original dataset only differentiates between America, Europe, and Asia). Front-wheel-drive cars have a lower consumption than rear-wheel-drive ones (the corresponding category being positively negatively with MPG at a level of 0.411), mostly due to the fact that they are lighter. Furthermore, a correlation with the car's design can be observed (e.g., hatchbacks having a lower consumption than station wagons). All those car characteristics are not included in the original dataset, but added from DBpedia.

### 7.3.2 Propositionalization Strategies

In [252] we performed an evaluation on different propositionalization strategies on three different data-mining tasks, i.e., classification, regression, and outlier detection, using three different data mining algorithms for each task. The evaluation was performed for the binary, numerical, relative count, and TF-IDF vector representation on five different feature sets. The evaluation showed that although the selected propositionalization strategy has a major impact on the data mining results, it is difficult to come up with a general recommendation for one strategy, as it depends on the given data mining task, the dataset at hand, and the data mining algorithm to be used.

[22] `http://archive.ics.uci.edu/ml/datasets/Auto+MPG`

Table 7.1: Feature Consolidation Results

| property | type | #sources | #fused sources | coverage | precision |
|---|---|---|---|---|---|
| population | numeric | 6 | 5 | 100.0% | 91.07% |
| area | numeric | 4 | 3 | 97.47% | 97.40% |
| currency | string | 2 | 2 | 96.86% | 97.40% |
| country code | string | 5 | 5 | 95.73% | 90.25% |

### 7.3.3 Feature Selection

In [253] we have performed an evaluation of the feature selection approach in hierarchical feature spaces, on six synthetic and five real world datasets, using three classification algorithms. Using our hierarchical feature selection approach, we were able to achieve a feature space compression of up to 95%, without decreasing the model performance, or in some cases even increasing it. The evaluation has shown that the approach outperforms standard feature selection techniques as well as recent hierarchy-aware approaches.

### 7.3.4 Data Integration

As discussed in our example in Section 7.2, our dataset contained $1,329$ attributes collected from ten different LOD sources. On that dataset, we first applied the *PARIS LOD Matcher* to detect the matching attributes in the dataset, which were later resolved using the *Data Fusion operator*.[23] After applying the operator, $858$ correspondences were discovered. The correspondences were resolved using the data fusion operator, using *majority voting* for strings and *median* for numeric values as resolution strategy. The fused dataset contained $1,090$ attributes.

We have evaluated both the matching and the fusion approach on four manually selected properties. First, for each of the properties, we manually counted the number of LOD sources that provide a value for at least one instance in the dataset. Then, we counted how many of those properties were matched and fused. The results are shown in Table 7.1, e.g., for the property *population* there are six LOD sources providing the values, but only five were matched and fused.

From the results, we can observe that the matching operator based on the PARIS approach is able to detect the matching properties with high accuracy. It only failed to detect one match for the *population* and *area* properties, which is the property from *Eurostat*. The reason for not detecting the matching property is that PARIS combines instance and schema level evidence. Since *Eurostat* provides mainly data for European countries, but the dataset at hand contains countries from all the world, there is not enough evidence that the property from the Eurostat should be matched to the properties from the other sources.

Furthermore, we evaluate the quality of the values of the fused features. For each instance, we manually retrieved the reference values for each property from

[23] For the PARIS LOD Matcher, we used the following parameters: *literal similarity distance = levenshtein*, *number of iteration = 10*, *alignment acceptance threshold = 0.2*, *post literal distance threshold = 0.58* and *literal normalization = true*.

Wikipedia. This is a common practice for such a task [36, 335], as there is no gold standard that can be used. The measures we use for our evaluation are *precision* and *coverage*. Coverage is the percentage of rows from the query table for which a value is provided. Precision is the percentage of correct values. We treat a numeric value as a correct match if it does not deviate more than $10\%$ from the reference value. Table 7.1 shows the resulting precision and coverage values.

The reason for a low precision for the property *country code* is that for some of the instances the values of the property differ across different sources, and the property has high sparsity in some of the sources, like DBpedia. Therefore, the majority voting strategy for such cases falls back to selecting the first non-null value provided from some of the sources, as there are no more than one occurrence of a value for the observed instance.

### 7.3.5 Time Performances

In this section we perform runtime evalution of the feature generation approaches, and the runtime for performing data analysis on the generated feature sets. All experiments were run using standard laptop computer with 4GB of RAM and Intel Core i7-3540M 3.0GHz CPU.

#### Scientific and technical journal articles

We measure the time for feature generation when using the complete LOD cloud, this includes discovering the *owl:sameAs* links and generating the data, the data integration, and the time for data analysis, i.e., correlation analysis. We do the same evaluation when using only DBpedia as a source. To measure the runtime for feature generation, we repeated the process three times, and report the average.

The results are depicted in Fig. 7.4. It can be observed that the largest portion of time is consumed by the feature generation step, which, in turn, is dominated by network access. Here, factors beyond the control of the user play an import role: reachability and responsiveness of the LOD endpoints used, network latencies, and intentional delays between requests to respect with access policies, which may impose maximum request rates.

All factors *not* influenced by factors originating in online data access, such as local performance of data analysis, can be addressed by design in RapidMiner, since there, e.g., are cloud computing services[24] as well as the Radoop extension[25] for data analysis with Hadoop, which allow for scaling the analytics operations.

#### Metacritic Movies

To further analyze the critical part, i.e., the feature generation, we have conducted a controlled scalability experiment, where we examine how the number of instances

[24] `https://rapidminer.com/products/cloud/`
[25] `https://rapidminer.com/products/radoop/`

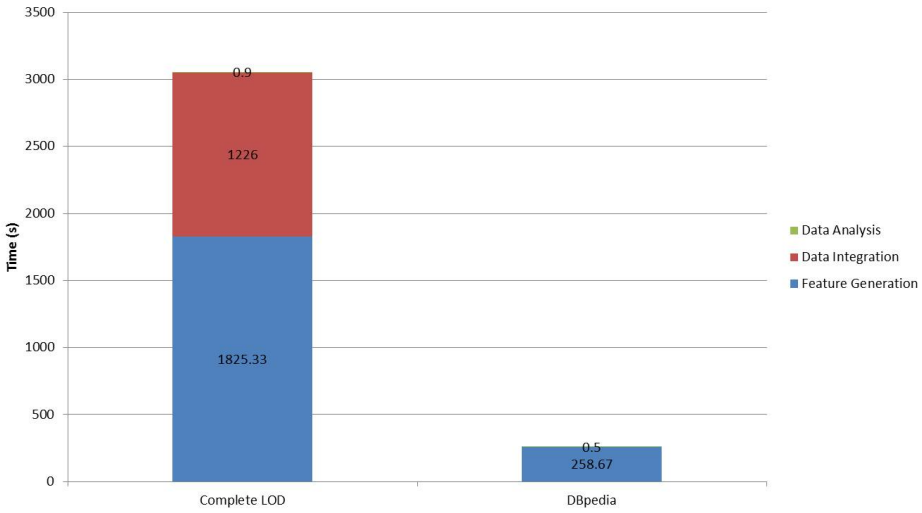



Figure 7.4: Runtime performances for feature generation, data integration and data analysis.

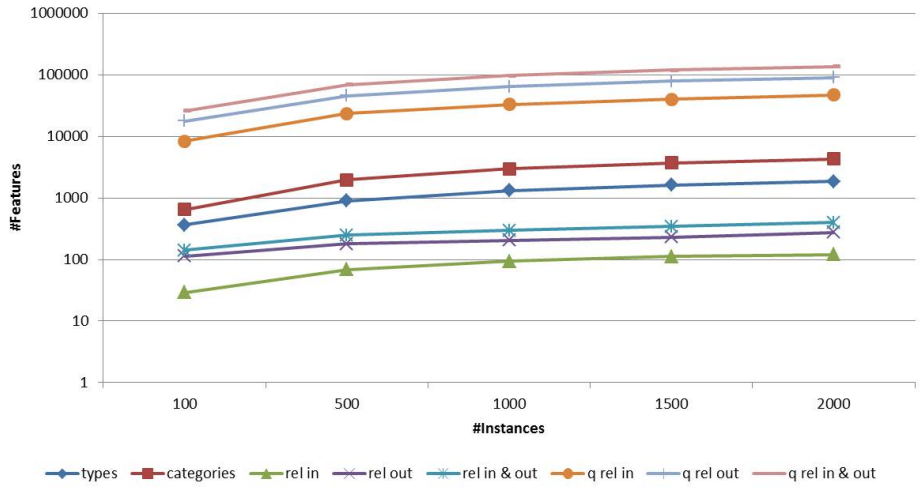



Figure 7.5: Features increase rate per strategy (log scale).

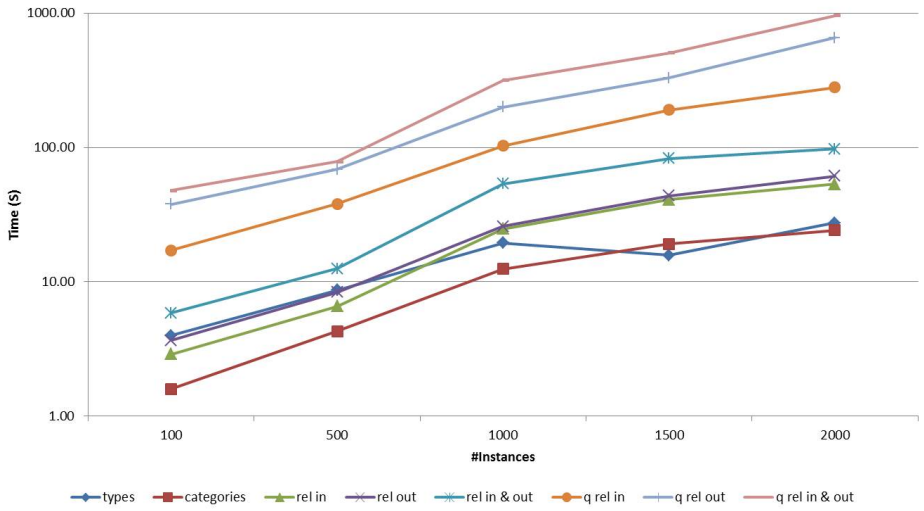



Figure 7.6: Feature generation runtime per strategy (log scale).

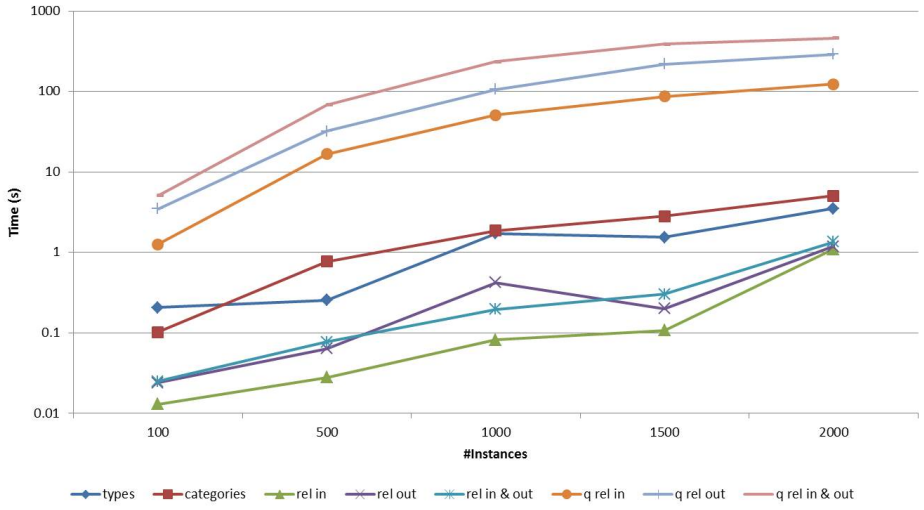



Figure 7.7: Naïve Bayes learning runtime (log scale).

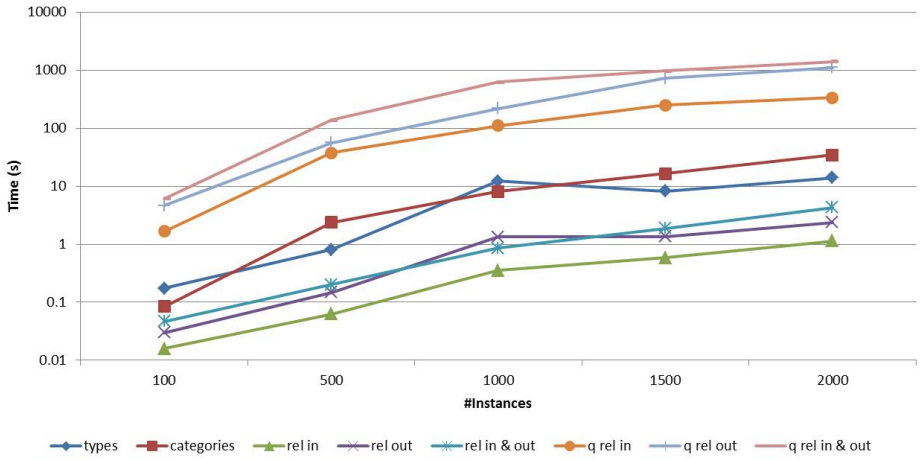



Figure 7.8: k-Nearest Neighbors learning runtime (log scale).

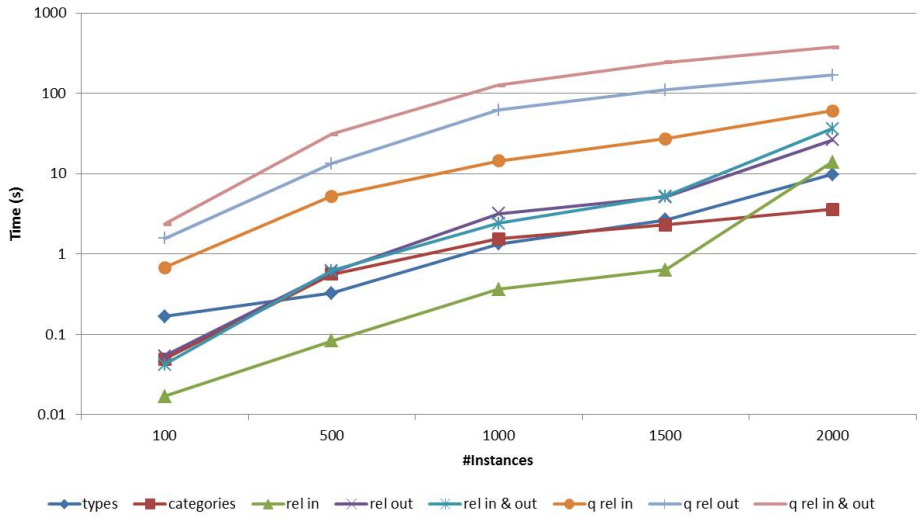



Figure 7.9: Support Vector Machines learning runtime (log scale).

affects the number of generated features by using types, Wikipedia categories, relations, and qualified relations as features (cf. section 7.1.3). We measure the time for generating the feature sets, and the time for learning three different classification methods, i.e., Naïve Bayes, k-Nearest Neighbors (with k=3), and Support Vector Machines. Since we are only interested in runtime measurements, not qualitative results of the data mining process, we do not perform parameter optimization. The runtime measures reflect the total time in seconds needed for performing 10-fold cross validation.

For the purpose of the evaluation, we use the *Metacritic Movies* dataset[26]. The *Metacritic Movies* dataset is retrieved from Metacritic.com[27], which contains an average rating of all time reviews for a list of movies. Each movie is linked to DBpedia using the movie's title and the movie's director. The initial dataset contained around $10,000$ movies, from which we selected $1,000$ movies from the top of the list, and $1,000$ movies from the bottom of the list. To use the dataset for classification, we discretize the target variable into "good" and "bad", using equal frequency binning. We perform the evaluation on five stratified sub-samples of the complete dataset with different sizes, i.e., $100$, $500$, $1,000$, $1,500$ and $2,000$.

In the results, we can see that there are three groups of generators, which show a different behavior. The smallest number of features are created by relations, a medium number by types and categories, and the largest number by qualified relations. However, as shown in Fig. 7.5, the number of features do not increase disproportionally as the number of instances is increased; and there is even a convergence to a stable state for most generators. The runtimes for feature generation also follow a linear increase, as shown in Fig. 7.6. Figures 7.7, 7.8, and 7.9 show that the runtimes for data analysis are likewise influenced by the number of features (and hence the number of instances), however, their contribution to the overall runtime is low, as discussed above.

## 7.4 Related Work

The use of Linked Open Data in data mining has been proposed before, and implementations as RapidMiner extensions as well as proprietary toolkits exist.

The direct predecessor of the RapidMiner LOD extension is the *FeGeLOD* toolkit [225], a data preprocessing toolkit based on the *Weka* platform [106], which contains basic versions of some of the operators offered by the LOD extension.

Different means to mine data in Linked Open Data sets have been proposed, e.g., an extension for RapidMiner [200], as well as standalone systems like *LiDDM* [140]. In those systems, data can be imported from public SPARQL endpoints using custom queries, but no means to join that data with local data are given. Cheng et al. [43] proposes an approach for automated feature generation after the user has

[26] http://data.dws.informatik.uni-mannheim.de/rmlod/LOD_ML_Datasets/data/datasets/MetacriticMovies/

[27] http://www.metacritic.com/browse/movies/score/metascore/all

specified the type of features. To do so, similar like the previous approaches, the users have to specify the SPARQL query, which makes this approach supervised rather than unsupervised. Mynarz et al. [192] have considered using user specified SPARQL queries in combination with SPARQL aggregates.

Similarly, the *RapidMiner SemWeb Extension* [145] is an extension for importing RDF from local files into RapidMiner, using custom SPARQL queries. As discussed above, RDF is a general graph format, which leads to the problem of set-valued features when transforming the data into the relational form used in RapidMiner. To cope with that issue, the extension provides different operators to transform the set-valued data into a lower-dimensional projection, which can be processed by standard RapidMiner operators.

Linked Open Data may also be loaded with the *RMOnto* [239] extension, which is similar to the SemWeb extension, but comes with a set of tailored relational learning algorithms and kernel functions. Together, these form a powerful package of operators, but it is difficult to combine them with built-in RapidMiner operators, as well as operators from other extensions.

Kauppinen et al. have developed the SPARQL package for R[28] [142], which allows importing LOD data in the well known environment for statistical computing *R*.

*Kernel functions* compute the distance between two data instances, by counting common substructures in the graphs of the instances, i.e. walks, paths and threes. Graph kernels are used in kernel-based data mining algorithms, e.g., support vector machines. In the past, many graph kernels have been proposed that are tailored towards specific application [128], or towards specific semantic representation [80]. Only few approaches are general enough to be applied on any given RDF data, regardless of the data mining task. Lösch et al. [169] introduce two general RDF graph kernels, based on intersection graphs and intersection trees. Later, the intersection tree path kernel was simplified by Vries et al. [58]. In another work, Vries et al. [57] introduce an approximation of the state-of-the-art Weisfeiler-Lehman graph kernel algorithm aimed at improving the computation time of the kernel when applied to RDF. Furthermore, the kernel implementation allows for explicit calculation of the instances' feature vectors, instead of pairwise similarities. Two variants of such kernel-based feature vectors are implemented in the Linked Open Data extension.

Tiddi et al. [300] introduced the *Dedalo* framework that traverses LOD to find commonalities that form explanations for items of a cluster. Given a supervised data mining task, such an approach could be easily adapted and used as feature generation approach.

All of those approaches miss a functionality to link local data to remote Linked Open Data, as in the use case discussed in section 7.2. Furthermore, they often require expert knowledge on Semantic Web technology, e.g., for formulating custom SPARQL queries, which is not necessary for our extension. Furthermore, auto-

[28] `http://linkedscience.org/tools/sparql-package-for-r/`

matic fusion of data from different sources is only scarcely supported.

The main use case for application discussed in this chapter, i.e., extending a given table with data from the web in a data analysis environment, has also been discussed for other web data than LOD. In RapidMiner, the *Finance and Economics Extension*[29] offers live access to stock market tickers and other data sources. For their Excel and BI products, Microsoft offers the *Power Query* extension, which also allows extending tables with tables from the web, and also offers means for data consolidation [326]. A publicly available prototype which provides for extending local tables with pre-crawled data from HTML tables and Microdata annotations in web pages as well as Linked Data is the *Mannheim Search Join Engine* [163].

## 7.5 Conclusion and Outlook

In this chapter, we have introduced the RapidMiner Linked Open Data extension. It provides a set of operators for augmenting existing datasets with additional attributes from open data sources, which often leads to better predictive and descriptive models. The RapidMiner Linked Open Data extension provides operators that allow for adding such attributes in an automatic, unsupervised manner.

There are different directions of research that are currently pursued in order to improve the extension. Besides developing new algorithms for the functionality already included (e.g., for linking and feature selection), there are also some new functionalities currently being investigated.

The current implementation is rather strict in its sequential process, i.e., it generates attributes first and filters and reconciles them later. Depending on the generation strategy used, this can lead to a large number of features being generated only to be discarded in the subsequent step. To avoid that overhead and improve the performance, we are working on mechanisms that decide on or estimate the utility of attribute already during creation and are capable of stopping the generation of attributes earlier if they seem to be useless.

Furthermore, more sophisticated propositionalization strategies might be developed. For example, the target variable from the local dataset can be used for developing supervised weighting approaches, as used for text mining in [98]. Furthermore, we can use the graph properties for calculating feature weights, e.g., the fan-in and fan-out values of the graph nodes can give a better representation of the popularity of the resources included in the features. Such a popularity score might be a good indicator of the feature's relevance for the data mining task. More sophisticated popularity scores can be calculated using some of the standard graph ranking algorithms, e.g., PageRank and HITS.

For feature selection, we have only regarded simple hierarchies so far. If features are organized in a complex ontology, there are other relations as well, which

[29] `https://marketplace.rapid-i.com/UpdateServer/faces/product_details.xhtml?productId=rmx_quantx1`

may be exploited for feature selection. Generalizing the approach to arbitrary relations between features is also a relevant direction of future work.

With respect to data integration, we have only used an off-the-shelf ontology matching approach to identify identical features across multiple LOD sources. In the future, more sophisticated matching approaches can be developed. Those should be able to handle data types beyond strings and numericals, i.e., dates, coordinates, and various units of measurement. Furthermore, the data source provenance information could be used in the fusion process to give more weight to data with higher quality and data that is up to date.

More information about the RapidMiner LOD extension, detailed user manual, and example processes can be found on the extension website[30]. The extension is available for free download from the RapidMiner marketplace[31] under the *AGPL* license[32].

[30] http://dws.informatik.uni-mannheim.de/en/research/rapidminer-lod-extension/

[31] https://marketplace.rapid-i.com/UpdateServer/faces/product_details.xhtml?productId=rmx_lod

[32] https://www.gnu.org/licenses/agpl-3.0.html

# Part II

# Semantic Web Knowledge Graphs Embeddings

# Chapter 8

# RDF2Vec: RDF Graph Embeddings for Data Mining

In the previous chapters of the thesis, we introduced several strategies for feature generation from Semantic Web knowledge graphs that can be used as background knowledge in various data mining tasks. However, we also saw that such feature strategies do not scale when the input dataset is large, i.e., the number of generated features quickly becomes unmanageable.

In this chapter, we introduce an approach for Semantic Web knowledge graphs embedding. In language modeling, vector space word embeddings have been proposed in 2013 by Mikolov et al. [177, 178]. They train neural networks for creating a low-dimensional, dense representation of words, which show two essential properties: (a) similar words are close in the vector space, and (b) relations between pairs of words can be represented as vectors as well, allowing for arithmetic operations in the vector space. In this work, we adapt those language modeling approaches for creating a latent representation of entities in RDF graphs. Since language modeling techniques work on sentences, we first convert the graph into a set of sequences of entities using two different approaches, i.e., graph walks and Weisfeiler-Lehman Subtree RDF graph kernels. In the second step, we use those sequences to train a neural language model, which estimates the likelihood of a sequence of entities appearing in a graph. Once the training is finished, each entity in the graph is represented as a vector of latent numerical features. We show that the properties of word embeddings also hold for RDF entity embeddings, and that they can be exploited for various tasks.

We use several RDF graphs to show that such latent representation of entities have high relevance for different data mining tasks. The generation of the entities' vectors is task and dataset independent, i.e., we show that once the vectors are generated, they can be used for machine learning tasks, like classification and regression. Furthermore, since all entities are represented in a low dimensional feature space, building the learning models and algorithms becomes more efficient.

**The work presented in this chapter has been published before as:" Petar Ristoski, Heiko Paulheim: *Rdf2Vec: RFG Graph Embeddings for Data Mining.* Proceedings of the 15th International Semantic Web Conference, Kobe, Japan, October, 2016."' [254], "Petar Ristoski, Jessica Rosati, Tommaso Di Noia, Renato de Leone, Heiko Paulheim: *RDF2Vec: RDF Graph Embeddings and Their Applications.* The Semantic Web Journal." [258]**

## 8.1 Approach

In our approach, we adapt neural language models for RDF graph embeddings. Such approaches take advantage of the word order in text documents, explicitly modeling the assumption that closer words in a sequence are statistically more dependent. In the case of RDF graphs, we consider entities and relations between entities instead of word sequences. Thus, in order to apply such approaches on RDF graph data, we first have to transform the graph data into sequences of entities, which can be considered as sentences. Using those sentences, we can train the same neural language models to represent each entity in the RDF graph as a vector of numerical values in a latent feature space.

### 8.1.1 RDF Graph Sub-Structures Extraction

We propose two general approaches for converting graphs into a set of sequences of entities, i.e, graph walks and Weisfeiler-Lehman Subtree RDF Graph Kernels. Following Definition 1, the objective of the conversion functions is for each vertex $v \in V$ to generate a set of sequences $S_v$, where the first token of each sequence $s \in S_v$ is the vertex $v$ followed by a sequence of tokens, which might be edge labels, vertex identifiers, or any substructure extracted from the RDF graph, in an order that reflects the relations between the vertex $v$ and the rest of the tokens, as well as among those tokens.

#### Graph Walks

In this approach, given a graph $G = (V, E)$, for each vertex $v \in V$, we generate all graph walks $P_v$ of depth $d$ rooted in vertex $v$. To generate the walks, we use the breadth-first algorithm. In the first iteration, the algorithm generates paths by exploring the direct outgoing edges of the root node $v_r$. The paths generated after the first iteration will have the following pattern $v_r \rightarrow e_i$, where $e_i \in E_{v_r}$, and $E_{v_r}$ is the set of all outgoing edges from the root node $v_r$. In the second iteration, for each of the previously explored edges the algorithm visits the connected vertices. The paths generated after the second iteration will follow the following pattern $v_r \rightarrow e_i \rightarrow v_i$. The algorithm continues until $d$ iterations are reached. The final set of sequences for the given graph $G$ is the union of the sequences of all the vertices $P_G = \bigcup_{v \in V} P_v$. The algorithm is shown in Algorithm 1.

**Algorithm 1:** Algorithm for generating RDF graph walks

**Data:** $G = (V, E)$: RDF Graph, $d$: walk depth
**Result:** $P_G$: Set of sequences

```
1  P_G = ∅
2  foreach vertex v ∈ V do
3      Q = initialize queue
4      w = initialize walk
5      add v to w
6      add Entry(v, w) to Q
7      while Q is nonempty do
8          entry = deq(Q)
9          currentVertex = entry.key
10         currentWalk = entry.value
11         if currentWalk.length == d then
12             add currentWalk to P_G
13             continue
14         end
15         E_c = currentVertex.outEdges()
16         foreach vertex e ∈ E_c do
17             w = currentWalk
18             add e to w
19             if w.length == d then
20                 add w to P_G
21                 continue
22             end
23             v_e = e.endVertex()
24             add v_e to w
25             add Entry(v_e, w) to Q
26         end
27     end
28 end
```

In the case of large RDF graphs, generating all possible walks for all vertices results in a large number of walks, which makes the training of the neural language model highly inefficient. To avoid this problem, we suggest for each vertex in the graph to generate only a subset, with size $n$, of all possible walks. To generate the walks, the outgoing edge to follow from the currently observed vertex $v_c$ is selected based on the edge weight, i.e., the probability for selecting an edge $e_i$ is $Pr[e_i] = \frac{weight(e_i)}{\sum_{j=1}^{|Ev_c|} weight(e_j)}$, where $e_i \in E_{v_c}$, and $E_{v_c}$ is the set of all outgoing edges from the current node $v_c$. While there are many possibilities to set the weight of the edges, in this work we only consider equal weights, i.e., random selection of outgoing edges where an edge $e_i$ is selected with probability $Pr[e_i] = \frac{1}{|E(v_c)|}$, where $e_i \in E_{v_c}$, and $E_{v_c}$ is the set of all outgoing edges from the current node $v_c$. The algorithm is shown in Algorithm 2. Other weighting strategies can be integrated into the algorithm by exchanging the function *selectEdge* in line 11, e.g., weighting the edge based on the frequency, based on the frequency of the edge's end node, or based on global weighting metrics, like PageRank [29].

**Weisfeiler-Lehman Subtree RDF Graph Kernels**

In this approach, we use the subtree RDF adaptation of the Weisfeiler-Lehman algorithm presented in [57, 59]. The Weisfeiler-Lehman Subtree graph kernel is a state-of-the-art, efficient kernel for graph comparison [277]. The kernel computes the number of sub-trees shared between two (or more) graphs by using the Weisfeiler-Lehman test of graph isomorphism. This algorithm creates labels representing subtrees in $h$ iterations. The rewriting procedure of Weisfeiler-Lehman goes as follows: (i) the algorithm creates a multiset label for each vertex based on the labels of the neighbors of that vertex; (ii) this multiset is sorted and together with the original label concatenated into a string, which is the new label; (iii) for each unique string a new (shorter) label replaces the original vertex label; (iv) at the end of each iteration, each label represents a unique full subtree.

There are two main modifications of the original Weisfeiler-Lehman graph kernel algorithm in order to be applicable on RDF graphs [57, 59]. First, the RDF graphs have directed edges, which is reflected in the fact that the neighborhood of a vertex $v$ contains only the vertices reachable via outgoing edges. Second, as mentioned in the original algorithm, labels from two iterations can potentially be different while still representing the same subtree. To make sure that this does not happen, the authors in [57, 59] have added tracking of the neighboring labels in the previous iteration, via the multiset of the previous iteration. If the multiset of the current iteration is identical to that of the previous iteration, the label of the previous iteration is reused.

The Weisfeiler-Lehman relabeling algorithm for an RDF graph is given in Algorithm 3, which is the same relabeling algorithm proposed in [57]. The algorithm takes as input the RDF graph $G = (V, E)$, a labeling function $l$, which returns a label of a vertex or edge in the graph based on an index, the subraph depth $d$

**Algorithm 2:** Algorithm for generating weighted RDF graph walks

**Data:** $G = (V, E)$: RDF Graph, $d$: walk depth, $n$: number of walks
**Result:** $P_G$: Set of sequences

$P_G := \emptyset$
**foreach** *vertex* $v \in V$ **do**
  $n_v$ = n
  **while** $n_v > 0$ **do**
    $w$ = initialize walk
    add $v$ to $w$
    $currentVertex = v$
    $d_v = d$
    **while** $d_v > 0$ **do**
      $E_c = currentVertex.outEdges()$
      $e$ = selectEdge($E_c$)
      $d_v = d_v$ - 1
      add $e$ to $w$
      **if** $d_v > 0$ **then**
        $v_e = e.endVertex()$
        add $v_e$ to $w$
        $currentVertex = v_e$
        $d_v = d_v$ - 1
      **end**
    **end**
    add $w$ to $P_G$
    $n_v = n_v$ - 1
  **end**
**end**

and the number of iterations $h$. The algorithm returns the labeling functions for each iteration $l_0$ to $l_h$, and a label dictionary $f$. Furthermore, the neighborhood $N(v) = (v', v) \in E$ of a vertex is the set of edges going to the vertex $v$ and the neighborhood $N((v, v')) = v$ of an edge is the vertex that the edge comes from.

The procedure of converting the RDF graph to a set of sequences of tokens goes as follows: (i) for a given graph $G = (V, E)$, we define the Weisfeiler-Lehman algorithm parameters, i.e., the number of iterations $h$ and the vertex subgraph depth $d$, which defines the subgraph in which the subtrees will be counted for the given vertex; (ii) after each iteration, for each vertex $v \in V$ of the original graph $G$, we extract all the paths of depth $d$ within the subgraph of the vertex $v$ on the relabeled graph using Algorithm 1. We set the original label of the vertex $v$ as the starting token of each path, which is then considered as a sequence of tokens. The sequences after each iteration will have the following pattern $v_r \rightarrow l_n(e_i, j) \rightarrow l_n(v_i, j)$, where $l_n$ returns the label of the edges and the vertices in the $n^{th}$ iteration. The sequecens could also be seen as $v_r \rightarrow T_1 \rightarrow T_1$ ... $T_d$, where $T_d$ is a subtree that appears on depth $d$ in the vertex's subgraph; (iii) we repeat step (ii) until the maximum iterations $h$ are reached. (iv) The final set of sequences is the union of the sequences of all the vertices in each iteration $P_G = \bigcup_{i=1}^{h} \bigcup_{v \in V} P_v$.

### 8.1.2 Neural Language Models – word2vec

Neural language models have been developed in the NLP field as an alternative to represent texts as a bag of words, and hence, a binary feature vector, where each vector index represents one word. While such approaches are simple and robust, they suffer from several drawbacks, e.g., high dimensionality and severe data sparsity, which limits their performance. To overcome such limitations, neural language models have been proposed, inducing low-dimensional, distributed embeddings of words by means of neural networks. The goal of such approaches is to estimate the likelihood of a specific sequence of words appearing in a corpus, explicitly modeling the assumption that closer words in the word sequence are statistically more dependent.

While some of the initially proposed approaches suffered from inefficient training of the neural network models, like Feedforward Neural Net Language Model (NNLM) [12, 47, 309], with the recent advances in the field several efficient approaches have been proposed. One of the most popular and widely used approaches is the word2vec neural language model [177, 178]. Word2vec is a particularly computationally-efficient two-layer neural net model for learning word embeddings from raw text. There are two different algorithms, the Continuous Bag-of-Words model (CBOW) and the Skip-gram model. The efficiency of the models comes as a result from the simplicity of the models by avoiding dense matrix multiplication, i.e., the non-linear hidden layer is removed from the neural network and the projection layer is shared for all words. Furthermore, the Skip-gram model has been extended to make the training even more efficient, i.e., (i) sub-sampling of frequent words, which significantly improves the model training efficiency, and

**Algorithm 3:** Weisfeiler-Lehman Relabeling for RDF

**Data:** $G = (V, E)$: RDF Graph, $l$: labeling function for G = (V, E), $d$: subgraph depth, $h$: number of iterations
**Result:** $l_0$ to $l_h$: label functions, $f$ label dictionary

**for** $n$ = *0;* $n < h$; *i++* **do**
  # 1. Multiset-label determination
  **foreach** $v \in V$ *and* $e \in E$ *and* $j = 0$ *to* $d$ **do**
    **if** $n = 0$ *and* $l(v, j)$ *is defined* **then**
      set $M_n(v, j) = l_0(v, j) = l(v, j)$
    **end**
    **if** $n = 0$ *and* $l(e, j)$ *is defined* **then**
      set $M_n(e, j) = l_0(e, j) = l(e, j)$
    **end**
    **if** $n > 0$ *and* $l(v, j)$ *is defined* **then**
      set $M_n(v, j) = \{l_{n-1}(u, j) | u \in N(v)\}$
    **end**
    **if** $n > 0$ *and* $l(e, j)$ *is defined* **then**
      set $M_n(e, j) = \{l_{n-1}(u, j + 1) | u \in N(e)\}$
    **end**
  **end**
  # 2. Sorting each multiset
  **foreach** $M_n(v, j)$ *and* $M_n(e, j)$ **do**
    sort the elements in $M_n(v, j)$, resp. $M_n(e, j)$, in ascending order and concatenate them into a string $s_n(v, j)$, resp. $s_n(e, j)$
  **end**
  **foreach** $s_n(v, j)$ *and* $s_n(e, j)$ **do**
    **if** $n > 0$ **then**
      add $l_{n-1}(v, j)$, resp. $l_{n-1}(e, j)$, as a prefix to $s_n(v, j)$, resp. $s_n(e, j)$
    **end**
  **end**
  # 3. Label compression
  **foreach** $s_n(v, j)$ *and* $s_n(e, j)$ **do**
    map $s_n(v, j)$, resp. $s_n(e, j)$, to a new compressed label, using a function $f : \sum^* \rightarrow \sum$, such that $f(s_n(v, j)) = f(s_n(v', j))$ iff $s_n(v, j) = s_n(v', j)$, resp. $f(s_n(e, j)) = f(s_n(e', j))$ iff $s_n(e, j) = s_n(e', j)$
  **end**
  # 4. Relabeling
  **foreach** $s_n(v, j)$ *and* $s_n(e, j)$ **do**
    set $l_n(v, j) = f(s_n(v, j))$ and $l_n(e, j) = f(s_n(e, j))$
  **end**
**end**

improves the vector quality of the less frequent words; (ii) using simplified variant of Noise Contrastive Estimation [105], called negative sampling.

### Continuous Bag-of-Words Model

The CBOW model predicts target words from context words within a given window. The model architecture is shown in Fig. 8.1a. The input layer is comprised of all the surrounding words for which the input vectors are retrieved from the input weight matrix, averaged, and projected in the projection layer. Then, using the weights from the output weight matrix, a score for each word in the vocabulary is computed, which is the probability of the word being a target word. Formally, given a sequence of training words $w_1, w_2, w_3, ..., w_T$, and a context window $c$, the objective of the CBOW model is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T} log\, p(w_t|w_{t-c}...w_{t+c}), \tag{8.1}$$

where the probability $p(w_t|w_{t-c}...w_{t+c})$ is calculated using the softmax function:

$$p(w_t|w_{t-c}...w_{t+c}) = \frac{exp(\bar{v}^T v'_{w_t})}{\sum_{w=1}^{V} exp(\bar{v}^T v'_w)}, \tag{8.2}$$

where $v'_w$ is the output vector of the word $w$, $V$ is the complete vocabulary of words, and $\bar{v}$ is the averaged input vector of all the context words:

$$\bar{v} = \frac{1}{2c}\sum_{-c\leq j\leq c, j\neq 0} v_{w_{t+j}} \tag{8.3}$$

### Skip-Gram Model

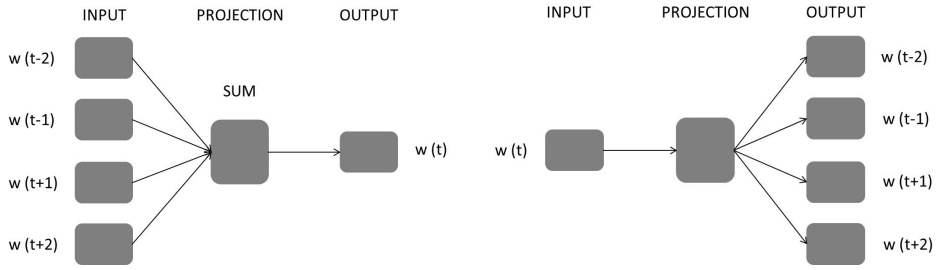
The skip-gram model does the inverse of the CBOW model and tries to predict the context words from the target words (Fig. 8.1b). More formally, given a sequence of training words $w_1, w_2, w_3, ..., w_T$, and a context window of size $c$, the objective of the skip-gram model is to maximize the following average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} log\, p(w_{t+j}|w_t), \tag{8.4}$$

where the probability $p(w_{t+j}|w_t)$ is calculated using the softmax function:

$$p(w_{t+c}|w_t) = \frac{exp(v'^T_{w_{t+c}} v_{w_t})}{\sum_{v=1}^{V} exp(v'^T_{w_v} v_{w_t})}, \tag{8.5}$$

where $v_w$ and $v'_w$ are the input and the output vector of the word $w$, and $V$ is the complete vocabulary of words.




(a) CBOW architecture (b) Skip-gram architecture

Figure 8.1: Architecture of the CBOW and Skip-gram model.

In both cases, calculating the softmax function is computationally inefficient, as the cost for computing is proportional to the size of the vocabulary. Therefore, two optimization techniques have been proposed, i.e., hierarchical softmax and negative sampling [178]. The empirical studies in the original paper [178] have shown that in most cases negative sampling leads to a better performance than hierarchical softmax, which depends on the selected negative samples, but it has higher runtime.

Once the training is finished, all words (or, in our case, entities) are projected into a lower-dimensional feature space, and semantically similar words (or entities) are positioned close to each other.

## 8.2 Evaluation

We evaluate our approach on two standard machine-learning tasks, i.e., classification and regression. Linking entities in a machine learning task to those in the LOD cloud helps generating additional features, which may help improving the overall learning outcome. For example, when learning a predictive model for the success of a movie, adding knowledge from the LOD cloud (such as the movie's budget, director, genre, Oscars won by the starring actors, etc.) can lead to a more accurate model.

We utilize two of the most prominent RDF knowledge graphs [223], i.e., DBpedia [162] and Wikidata [319]. DBpedia is a knowledge graph which is extracted from structured data in Wikipedia. The main source for this extraction are the key-value pairs in the Wikipedia infoboxes. Wikidata is a collaboratively edited knowledge graph, operated by the Wikimedia foundation[1] that also hosts various language editions of Wikipedia.

We use the English version of the 2015-10 DBpedia dataset, which contains $4,641,890$ instances and $1,369$ mapping-based object properties[2]. In our evalua-

[1] http://wikimediafoundation.org/

[2] http://wiki.dbpedia.org/services-resources/datasets/dbpedia-datasets

tion, we only consider object properties, and ignore datatype properties and literals.

For the Wikidata dataset, we use the simplified and derived RDF dumps from 2016-03-28[3]. The dataset contains $17,340,659$ entities in total. As for the DBpedia dataset, we only consider object properties, and ignore the data properties and literals.

The first step of our approach is to convert the RDF graphs into a set of sequences. As the number of generated walks increases exponentially [59] with the graph traversal depth, calculating Weisfeiler-Lehman subtrees RDF kernels, or all graph walks with a given depth $d$ for all of the entities in the large RDF graph quickly becomes unmanageable. Therefore, to extract the entities embeddings for the large RDF datasets, we use only random graph walks entity sequences, generated using Algorithm 2. For both DBpedia and Wikidata, we first experiment with 200 random walks per entity with depth of 4, and 200 dimensions for the entities' vectors. Additionally, for DBpedia we experiment with 500 random walks per entity with depth of 4 and 8, with 200 and 500 dimensions for the entities' vectors. For Wikidata, we were unable to build models with more than 200 walks per entity, because of memory constrains, therefore we only experiment with the dimensions of the entities' vectors, i.e., 200 and 500.

We use the corpora of sequences to build both CBOW and Skip-Gram models with the following parameters: window size = 5; number of iterations = 5; negative sampling for optimization; negative samples = 25; with average input vector for CBOW. The parameter values are selected based on recommendations from the literature [177]. To prevent sharing the context between entities in different sequences, each sequence is considered as a separate input in the model, i.e., the sliding window restarts for each new sequence. We used the *gensim* implementation[4] for training the models. All the models, as well as the code, are publicly available.[5]

In the evaluation section we use the following notation for the models: *KB2Vec model #walks #dimensions depth*, e.g. *DB2vec SG 200w 200v 4d*, refers to a model built on DBpedia using the skip-gram model, with 200 walks per entity, 200 dimensional vectors and all the walks are of depth 4.

## 8.3 Experimental Setup

For evaluating the performance of our RDF embeddings in machine learning tasks, we perform an evaluation on a set of benchmark datasets. The dataset contains three smaller-scale RDF datasets (i.e., AIFB, MUTAG, and BGS), where the classification target is the value of a selected property within the dataset, and five larger datasets linked to DBpedia and Wikidata, where the target is an external variable

[3] http://tools.wmflabs.org/wikidata-exports/rdf/index.php?content=dump_download.php&dump=20160328

[4] https://radimrehurek.com/gensim/

[5] http://data.dws.informatik.uni-mannheim.de/rdf2vec/

Table 8.1: Datasets overview. For each dataset, we depict the number of instances, the machine learning tasks in which the dataset is used (*C* stands for classification, and *R* stands for regression) and the source of the dataset. In case of classification, *c* indicates the number of classes.

| Dataset | #Instances | ML Task | Original Source |
|---|---|---|---|
| Cities | 212 | R/C (c=3) | Mercer |
| Metacritic Albums | 1600 | R/C (c=2) | Metacritic |
| Metacritic Movies | 2000 | R/C (c=2) | Metacritic |
| AAUP | 960 | R/C (c=3) | JSE |
| Forbes | 1585 | R/C (c=3) | Forbes |
| AIFB | 176 | C (c=4) | AIFB |
| MUTAG | 340 | C (c=2) | MUTAG |
| BGS | 146 | C (c=2) | BGS |

(e.g., the metacritic score of an album or a movie). The latter datasets are used both for classification and regression. Details on the datasets can be found in [247].

For each of the small RDF datasets, we first build two corpora of sequences, i.e., the set of sequences generated from graph walks with depth 8 (marked as W2V), and set of sequences generated from Weisfeiler-Lehman subtree kernels (marked as K2V). For the Weisfeiler-Lehman algorithm, we use 3 iterations and depth of 4, and after each iteration we extract all walks for each entity with the same depth. We use the corpora of sequences to build both CBOW and Skip-Gram models with the following parameters: window size = 5; number of iterations = 10; negative sampling for optimization; negative samples = 25; with average input vector for CBOW. We experiment with 200 and 500 dimensions for the entities' vectors.

We use the RDF embeddings of DBpedia and Wikidata (see Section 9.2) on the five larger datasets, which provide classification/regression targets for DBpedia/Wikidata entities (see Table 11.1).

We compare our approach to several baselines. For generating the data mining features, we use three strategies that take into account the direct relations to other resources in the graph [225, 252], and two strategies for features derived from graph sub-structures [59]:

- Features derived from specific relations. In the experiments we use the relations *rdf:type* (types), and *dcterms:subject* (categories) for datasets linked to DBpedia.
- Features derived from generic relations, i.e., we generate a feature for each incoming (rel in) or outgoing relation (rel out) of an entity, ignoring the value or target entity of the relation. Furthermore, we combine both incoming and outgoing relations (rel in & out).
- Features derived from generic relations-values, i.e., we generate a feature for each incoming (rel-vals in) or outgoing relation (rel-vals out) of an entity including the value of the relation. Furthermore, we combine both incoming and

outgoing relations with the values (rel-vals in & out).

- Kernels that count substructures in the RDF graph around the instance node. These substructures are explicitly generated and represented as sparse feature vectors.

  - The Weisfeiler-Lehman (WL) graph kernel for RDF [59] counts full subtrees in the subgraph around the instance node. This kernel has two parameters, the subgraph depth $d$ and the number of iterations $h$ (which determines the depth of the subtrees). We use two pairs of settings, $d = 2, h = 2$ (WL_2_2) and $d = 4, h = 3$ (WL_4_3).
  - The Intersection Tree Path kernel for RDF [59] counts the walks in the subtree that spans from the instance node. Only the walks that go through the instance node are considered. We will therefore refer to it as the root Walk Count (WC) kernel. The root WC kernel has one parameter: the length of the paths $l$, for which we test $4$ (WC_4) and $6$ (WC_6).

Furthermore, we compare the results to the state-of-the art graph embeddings approaches: TransE, TransH and TransR. These approaches have shown comparable results with the rest of the graph embeddings approaches on the task of link predictions. But most importantly, while there are many graph embeddings approaches, like RESCAL [198], Neural Tensor Networks (NTN) [281], ComplEx [307], HolE [197] and others, the approaches based on translating embeddings approaches scale to large knowledge-graphs as DBpedia.[6] We use an existing implementation and build models on the small RDF datasets and the whole DBpedia data with the default parameters.[7] For all the models we train $1,000$ epochs and build vectors with size $100$. We have to note that the primary goal of such embeddings is the link prediction task, not standard machine learning tasks.

For all the experiments we don't perform any feature selection, i.e., we use all the features generated with the given feature generation strategy. For the baseline feature generation strategies, we use binary feature vectors, i.e., 1 if the feature exists for the instance, 0 otherwise.

We perform two learning tasks, i.e., classification and regression. For classification tasks, we use Naive Bayes, k-Nearest Neighbors (k=3), C4.5 decision tree, and Support Vector Machines (SVMs). For the SVM classifier we optimize the complexity constant *C*[8] in the range $\{10^{-3}, 10^{-2}, 0.1, 1, 10, 10^2, 10^3\}$. For regression, we use Linear Regression, M5Rules, and k-Nearest Neighbors (k=3). We measure accuracy for classification tasks, and root mean squared error (RMSE) for regression tasks. The results are calculated using stratified 10-fold cross validation.

[6] Because of high processing requirements we were not able to build the models for the Wikidata dataset.

[7] `https://github.com/thunlp/KB2E/`

[8] The complexity constant sets the tolerance for misclassification, where higher C values allow for "softer" boundaries and lower values create "harder" boundaries.

The strategies for creating propositional features from Linked Open Data are implemented in the RapidMiner LOD extension[9] [229, 246]. The experiments, including the feature generation and the evaluation, were performed using the RapidMiner data analytics platform.[10] The RapidMiner processes and the complete results can be found online.[11]. The experiments were run using a Linux machine with 20GB RAM and 4 Intel Xeon 2.60GHz CPUs.

## 8.4 Results

The results for the task of classification on the small RDF datasets are shown in Table 8.2.[12] Experiments marked with "\" did not finish within ten days, or have run out of memory. The reason for that is the high number of generated features for some of the strategies, as explained in Section 8.6. From the results we can observe that the K2V approach outperforms all the other approaches. More precisely, using the skip-gram feature vectors of size 500 in an SVM model provides the best results on all three datasets. The W2V approach on all three datasets performs closely to the standard graph substructure feature generation strategies, but it does not outperform them. K2V outperforms W2V because it is able to capture more complex substructures in the graph, like sub-trees, while W2V focuses only on graph paths. Furthermore, the related approaches, perform rather well on the AIFB dataset, and achieve comparable results to the K2V approach, however, on the other two dataset K2V significantly outperforms all three of them.

The results for the task of classification on the five datasets using the DBpedia and Wikidata entities' vectors are shown in Table 8.3, and the results for the task of regression on the five datasets using the DBpedia and Wikidata entities' vectors are given in Table 8.4. We can observe that the latent vectors extracted from DBpedia and Wikidata outperform all of the standard feature generation approaches. Furthermore, the RDF2vec approaches built on the DBpedia dataset continuously outperform the related approaches, i.e., TransE, TransH, and TransR, on both tasks for all the datasets, except on the Forbes dataset for the task of classification. In this case, all the related approaches outperform the baseline approaches as well as the RDF2vec approach. The difference is most significant when using the C4.5 classifier. In general, the DBpedia vectors work better than the Wikidata vectors, where the skip-gram vectors with size 200 or 500 built on graph walks of depth 8 on most of the datasets lead to the best performances. An exception is the AAUP dataset, where the Wikidata skip-gram 500 vectors outperform the other approaches. The reason for this could be that Wikidata contains more information about universities.

[9] http://dws.informatik.uni-mannheim.de/en/research/rapidminer-lod-extension

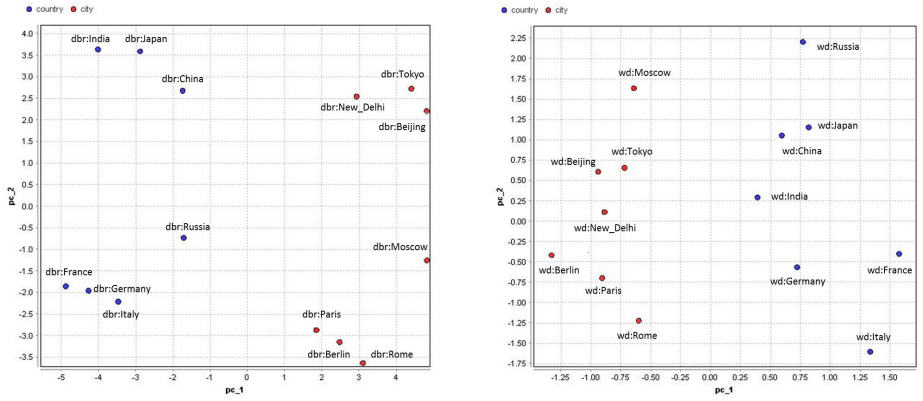
[10] https://rapidminer.com/

[11] http://data.dws.informatik.uni-mannheim.de/rmlod/LOD_ML_Datasets/

[12] We do not consider the strategies for features derived from specific relations, i.e., types and categories, because the datasets do not contain categories, and all the instances are of the same type

Table 8.2: Classification results on the small RDF datasets. The best results are marked in bold. Experiments marked with "\" did not finish within ten days, or have run out of memory.

| Strategy/Dataset | AIFB | | | | MUTAG | | | | BGS | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NB | KNN | SVM | C4.5 | NB | KNN | SVM | C4.5 | NB | KNN | SVM | C4.5 |
| rel in | 16.99 | 47.19 | 50.70 | 50.62 | \ | \ | \ | \ | 61.76 | 54.67 | 63.76 | 63.76 |
| rel out | 45.07 | 45.56 | 50.70 | 51.76 | 41.18 | 54.41 | 62.94 | 62.06 | 54.76 | 69.05 | 72.70 | 69.33 |
| rel in & out | 25.59 | 51.24 | 50.80 | 51.80 | \ | \ | \ | \ | 54.76 | 67.00 | 72.00 | 70.00 |
| rel-vals in | 73.24 | 54.54 | 81.86 | 80.75 | \ | \ | \ | \ | 79.48 | 83.52 | 86.50 | 68.57 |
| rel-vals out | 86.86 | 55.69 | 82.39 | 71.73 | 62.35 | 62.06 | 73.53 | 62.94 | 84.95 | 65.29 | 83.10 | 73.38 |
| rel-vals in&out | 87.42 | 57.91 | 88.57 | 85.82 | \ | \ | \ | \ | 84.95 | 70.81 | 85.80 | 72.67 |
| WL_2_2 | 85.69 | 53.30 | 92.68 | 71.08 | 91.12 | 62.06 | 92.59 | 93.29 | 85.48 | 63.62 | 82.14 | 75.29 |
| WL_4_3 | 85.65 | 65.95 | 83.43 | 89.25 | 70.59 | 62.06 | 94.29 | 93.47 | 90.33 | 85.57 | 91.05 | 87.67 |
| WC_4 | 86.24 | 60.27 | 75.03 | 71.05 | 90.94 | 62.06 | 91.76 | 93.82 | 84.81 | 69.00 | 83.57 | 76.90 |
| WC_6 | 86.83 | 64.18 | 82.97 | 71.05 | 92.00 | 72.56 | 86.47 | 93.82 | 85.00 | 67.00 | 78.71 | 76.90 |
| TransE | 82.29 | 90.85 | 89.74 | 62.94 | 72.65 | 47.65 | 72.65 | 65.29 | 61.52 | 70.67 | 65.71 | 63.57 |
| TransH | 80.65 | 88.10 | 84.67 | 59.74 | 70.59 | 43.24 | 70.29 | 57.06 | 58.81 | 69.95 | 69.38 | 58.95 |
| TransR | 80.03 | 90.26 | 89.74 | 58.53 | 72.35 | 46.76 | 72.94 | 59.12 | 61.33 | 61.76 | 64.43 | 57.48 |
| W2V CBOW 200 | 70.00 | 69.97 | 79.48 | 65.33 | 74.71 | 72.35 | 80.29 | 74.41 | 56.14 | 74.00 | 74.71 | 67.38 |
| W2V CBOW 500 | 69.97 | 69.44 | 82.88 | 73.40 | 75.59 | 70.59 | 82.06 | 72.06 | 55.43 | 73.95 | 74.05 | 65.86 |
| W2V SG 200 | 76.76 | 71.67 | 87.39 | 65.36 | 70.00 | 71.76 | 77.94 | 68.53 | 66.95 | 69.10 | 75.29 | 71.24 |
| W2V SG 500 | 76.67 | 76.18 | 89.55 | 71.05 | 72.35 | 72.65 | 78.24 | 68.24 | 68.38 | 71.19 | 78.10 | 63.00 |
| K2V CBOW 200 | 85.16 | 84.48 | 87.48 | 76.08 | 78.82 | 69.41 | 86.47 | 68.53 | 93.14 | 95.57 | 94.71 | 88.19 |
| K2V CBOW 500 | 90.98 | 88.17 | 86.83 | 76.18 | 80.59 | 70.88 | 90.88 | 66.76 | 93.48 | 95.67 | 94.82 | 87.26 |
| K2V SG 200 | 85.65 | 87.96 | 90.82 | 75.26 | 78.53 | 69.29 | 95.88 | 66.00 | 91.19 | 93.24 | 95.95 | 87.05 |
| K2V SG 500 | 88.73 | 88.66 | **93.41** | 69.90 | 82.06 | 70.29 | **96.18** | 66.18 | 91.81 | 93.19 | **96.33** | 80.76 |




(a) DBpedia vectors

(b) Wikidata vectors

Figure 8.2: Two-dimensional PCA projection of the 500-dimensional Skip-gram vectors of countries and their capital cities.

On both tasks, we can observe that the skip-gram vectors perform better than the CBOW vectors. Furthermore, the vectors with higher dimensionality and longer walks lead to a better representation of the entities and better performance on most of the datasets. However, for the variety of tasks at hand, there is no universal approach, i.e., a combination of an embedding model and a machine learning method, that consistently outperforms the others.

## 8.5 Semantics of Vector Representations

To analyze the semantics of the vector representations, we employ Principal Component Analysis (PCA) to project the entities' feature vectors into a two dimensional feature space. We selected seven countries and their capital cities, and visualized their vectors as points in a two-dimensional space. Figure 8.2a shows the corresponding DBpedia vectors, and Figure 8.2b shows the corresponding Wikidata vectors. The figure illustrates the ability of the model to automatically organize entities of different types, and preserve the relationships between different entities. For example, we can see that there is a clear separation between the countries and the cities, and the relation "capital" between each pair of country and the corresponding capital city is preserved. Furthermore, we can observe that more similar entities are positioned closer to each other, e.g., we can see that the countries that are part of the EU are closer to each other, and the same applies for the Asian countries.

Table 8.3: Classification results. The first number represents the dimensionality of the vectors, while the second number represent the value for the depth parameter. The best results are marked in bold. Experiments marked with "\" did not finish within ten days, or have run out of memory.

| Strategy/Dataset | Cities | | | | Metacritic Movies | | | | Metacritic Albums | | | | AAUP | | | | Forbes | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NB | KNN | SVM | C4.5 | NB | KNN | SVM | C4.5 | NB | KNN | SVM | C4.5 | NB | KNN | SVM | C4.5 | NB | KNN | SVM | C4.5 |
| types | 55.71 | 56.17 | 63.21 | 59.05 | 68.00 | 57.60 | 71.40 | 70.00 | 66.50 | 50.75 | 62.31 | 54.44 | 41.00 | 85.62 | 91.67 | 92.78 | 55.08 | 75.84 | 75.67 | 75.85 |
| categories | 55.74 | 49.98 | 62.39 | 56.17 | 75.25 | 62.70 | 76.35 | 69.50 | 67.40 | 54.13 | 64.50 | 56.62 | 48.00 | 85.83 | 90.78 | 91.87 | 60.38 | 76.11 | 75.70 | 75.70 |
| rel in | 60.41 | 58.46 | 71.70 | 60.35 | 52.75 | 49.90 | 60.35 | 60.10 | 51.13 | 62.19 | 65.25 | 60.75 | 45.63 | 85.94 | 90.62 | 92.81 | 50.24 | 76.49 | 75.16 | 76.10 |
| rel out | 47.62 | 60.00 | 66.04 | 56.71 | 52.90 | 58.45 | 66.40 | 62.70 | 58.75 | 63.75 | 62.25 | 64.50 | 41.15 | 85.83 | 89.58 | 91.35 | 64.73 | 75.84 | 75.73 | 75.92 |
| rel in & out | 59.44 | 58.57 | 66.04 | 56.47 | 52.95 | 59.30 | 67.75 | 62.55 | 58.69 | 64.50 | 67.38 | 61.56 | 42.71 | 85.94 | 89.67 | 92.50 | 22.27 | 75.96 | 76.34 | 75.98 |
| rel-vals in | \ | \ | \ | \ | 50.60 | 50.00 | 50.60 | 50.00 | 50.88 | 50.00 | 50.81 | 50.00 | 54.06 | 84.69 | 89.51 | \ | 14.95 | 76.15 | 76.97 | 75.73 |
| rel-vals out | 53.79 | 35.91 | 55.66 | 64.13 | 78.50 | 54.78 | 78.71 | \ | 74.06 | 52.56 | 76.99 | \ | 57.81 | 85.73 | 91.46 | 91.78 | 67.09 | 75.61 | 75.74 | 76.74 |
| rel-vals in&out | \ | \ | \ | \ | 77.90 | 55.75 | 77.82 | \ | 74.25 | 51.25 | 75.85 | \ | 63.44 | 84.69 | 91.56 | \ | 67.20 | 75.88 | 75.96 | 76.75 |
| WL_2_2 | 70.98 | 49.31 | 65.34 | 75.29 | 75.45 | 66.90 | 79.30 | 70.80 | 73.63 | 64.69 | 76.25 | 62.00 | 58.33 | 91.04 | 91.46 | 92.40 | 64.17 | 75.71 | 75.10 | 76.59 |
| WL_4_3 | 65.48 | 53.29 | 69.90 | 69.31 | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ |
| WC_4 | 72.71 | 47.39 | 66.48 | 75.13 | 75.39 | 65.89 | 74.93 | 69.08 | 72.00 | 60.63 | 76.88 | 63.69 | 57.29 | 90.63 | 93.44 | 92.60 | 64.23 | 75.77 | 76.22 | 76.47 |
| WC_6 | 65.52 | 52.36 | 67.95 | 65.15 | 74.25 | 55.30 | 78.40 | \ | 72.81 | 52.87 | 77.94 | \ | 57.19 | 90.73 | 90.94 | 92.60 | 64.04 | 75.65 | 76.22 | 76.59 |
| DB_TransE | 65.79 | 75.71 | 74.63 | 61.50 | 65.75 | 64.17 | 68.96 | 61.16 | 62.81 | 60.48 | 64.17 | 56.86 | 80.28 | 84.86 | 28.95 | 89.65 | 92.88 | 79.98 | 74.37 | **95.44** |
| DB_TransH | 64.39 | 72.66 | 76.66 | 60.89 | 63.51 | 63.25 | 67.43 | 60.96 | 63.97 | 63.13 | 65.07 | 60.23 | 80.39 | 84.86 | 27.55 | 89.21 | 93.82 | 79.98 | 74.37 | 93.68 |
| DB_TransR | 63.08 | 67.32 | 74.50 | 59.84 | 64.38 | 60.16 | 64.43 | 52.04 | 63.56 | 59.68 | 66.41 | 60.39 | 79.19 | 84.86 | 28.95 | 89.00 | 93.28 | 79.98 | 74.37 | 93.70 |
| DB2vec SG 200w 200v 4d | 75.66 | 73.50 | 75.68 | 47.58 | 77.58 | 78.24 | 79.61 | 72.78 | 73.85 | 73.79 | 75.28 | 65.95 | 78.21 | 85.18 | 29.57 | 91.83 | 82.24 | 81.07 | 74.10 | 85.23 |
| DB2vec CBOW 200w 200v 4d | 68.45 | 68.39 | 71.53 | 60.34 | 70.64 | 72.83 | 75.43 | 69.11 | 70.03 | 65.31 | 73.53 | 60.84 | 73.09 | 85.29 | 29.57 | 91.40 | 87.34 | 80.38 | 74.10 | 84.40 |
| DB2vec CBOW 500w 200v 4d | 59.32 | 68.84 | 77.39 | 64.32 | 65.60 | 79.74 | 82.90 | 74.33 | 70.72 | 71.86 | 76.36 | 67.24 | 73.36 | 89.65 | 29.00 | 92.45 | 89.38 | 80.94 | 76.83 | 84.81 |
| DB2vec CBOW 500w 500v 4d | 59.32 | 71.34 | 76.37 | 66.34 | 65.65 | 79.49 | 82.75 | 73.87 | 69.71 | 71.93 | 75.41 | 65.65 | 72.71 | 89.65 | 29.11 | 92.01 | 89.02 | 80.82 | 76.95 | 85.17 |
| DB2vec SG 500w 200v 4d | 60.34 | 71.82 | 76.37 | 65.37 | 65.25 | 80.44 | 83.25 | 73.87 | 68.95 | 73.89 | 76.11 | 67.87 | 71.20 | 89.65 | 28.90 | 92.12 | 88.78 | 80.82 | 77.92 | 85.77 |
| DB2vec SG 500w 500v 4d | 58.34 | 72.84 | 76.87 | 67.84 | 65.45 | 80.14 | **83.65** | 72.82 | 70.41 | 74.34 | 78.44 | 67.49 | 71.19 | 89.65 | 28.90 | 92.23 | 88.30 | 80.94 | 77.25 | 84.81 |
| DB2vec CBOW 500w 200v 8d | 69.26 | 69.87 | 67.32 | 63.13 | 57.83 | 70.08 | 65.25 | 67.47 | 67.91 | 64.44 | 72.42 | 65.39 | 68.18 | 85.33 | 28.90 | 90.50 | 77.35 | 80.34 | 28.90 | 85.17 |
| DB2vec CBOW 500w 500v 8d | 62.26 | 69.87 | 76.84 | 63.21 | 58.78 | 69.82 | 69.46 | 67.67 | 67.53 | 65.83 | 74.26 | 63.42 | 62.90 | 85.22 | 29.11 | 90.61 | 89.86 | 80.34 | 78.65 | 84.81 |
| DB2vec SG 500w 200v 8d | 73.32 | 75.89 | 78.92 | 60.74 | 79.94 | 79.49 | 83.30 | 75.13 | 77.25 | 76.87 | **79.72** | 69.14 | 78.53 | 85.12 | 29.22 | 91.04 | 90.10 | 80.58 | 78.96 | 84.68 |
| DB2vec SG 500w 500v 8d | **89.73** | 69.16 | 84.19 | 72.25 | 80.24 | 78.68 | 82.80 | 72.42 | 73.57 | 76.30 | 78.20 | 68.70 | 75.07 | **94.48** | 29.11 | 94.15 | 88.53 | 80.58 | 77.79 | 86.38 |
| WD2vec CBOW 200w 200v 4d | 68.76 | 57.71 | 75.56 | 61.37 | 51.49 | 52.20 | 51.64 | 49.01 | 50.86 | 50.29 | 51.44 | 50.09 | 50.54 | 90.18 | 89.63 | 88.83 | 49.84 | 81.08 | 76.77 | 79.14 |
| WD2vec CBOW 200w 500v 4d | 68.24 | 57.75 | 85.56 | 64.54 | 49.22 | 48.56 | 51.04 | 50.98 | 53.08 | 50.03 | 52.33 | 53.28 | 48.45 | 90.39 | 89.74 | 88.31 | 51.95 | 80.74 | 78.18 | 80.32 |
| WD2vec SG 200w 200v 4d | 72.58 | 57.53 | 75.48 | 52.32 | 69.53 | 70.14 | 75.39 | 67.00 | 60.32 | 62.03 | 64.76 | 58.54 | 60.87 | 90.50 | 89.63 | 89.98 | 65.45 | 81.17 | 77.74 | 77.03 |
| WD2vec SG 200w 500v 4d | 83.20 | 60.72 | 79.87 | 61.67 | 71.10 | 70.19 | 76.30 | 67.31 | 55.31 | 58.92 | 63.42 | 56.63 | 55.85 | 90.60 | 89.63 | 87.69 | 58.95 | 81.17 | 79.00 | 79.56 |

Table 8.4: Regression results. The first number represents the dimensionality of the vectors, while the second number represent the value for the depth parameter. The best results are marked in bold. Experiments that did not finish within ten days, or that have run out of memory are marked with "\".

| Strategy/Dataset | Cities | | | Metacritic Movies | | | Metacritic Albums | | | AAUP | | | Forbes | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LR | KNN | M5 | LR | KNN | M5 | LR | KNN | M5 | LR | KNN | M5 | LR | KNN | M5 |
| types | 24.30 | 22.16 | 18.79 | 77.80 | 30.68 | 22.16 | 16.45 | 18.36 | 13.95 | 9.83 | 34.95 | 6.28 | 29.22 | 21.07 | 18.32 |
| categories | 18.88 | 22.68 | 22.32 | 84.57 | 23.87 | 22.50 | 16.73 | 16.64 | 13.95 | 8.08 | 34.94 | 6.16 | 19.16 | 21.48 | 18.39 |
| rel in | 49.87 | 18.53 | 19.21 | 22.60 | 41.40 | 22.56 | 13.50 | 22.06 | 13.43 | 9.69 | 34.98 | 6.56 | 27.56 | 20.93 | 18.60 |
| rel out | 49.87 | 18.53 | 19.21 | 21.45 | 24.42 | 20.74 | 13.32 | 14.59 | 13.06 | 8.82 | 34.95 | 6.32 | 21.73 | 21.11 | 18.97 |
| rel in & out | 40.80 | 18.21 | 18.80 | 21.45 | 24.42 | 20.74 | 13.33 | 14.52 | 12.91 | 12.97 | 34.95 | 6.36 | 26.44 | 20.98 | 19.54 |
| rel-vals in | \ | \ | \ | 21.46 | 24.19 | 20.43 | 13.94 | 23.05 | 13.95 | \ | 34.96 | 6.27 | \ | 20.86 | 19.31 |
| rel-vals out | 20.93 | 23.87 | 20.97 | 25.99 | 32.18 | 22.93 | \ | 15.28 | 13.34 | \ | 34.95 | 6.18 | \ | 20.48 | 18.37 |
| rel-vals in&out | \ | \ | \ | \ | 25.37 | 20.96 | \ | 15.47 | 13.33 | \ | 34.94 | 6.18 | \ | 20.20 | 18.20 |
| WL_2_2 | 20.21 | 24.60 | 20.85 | \ | 21.62 | 19.84 | \ | 13.99 | 12.81 | \ | 34.96 | 6.27 | \ | 19.81 | 19.49 |
| WL_4_3 | 17.79 | 20.42 | 17.04 | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ |
| WC_4 | 20.33 | 25.95 | 19.55 | \ | 22.80 | 22.99 | \ | 14.54 | 12.87 | 9.12 | 34.95 | 6.24 | \ | 20.45 | 19.26 |
| WC_6 | 19.51 | 33.16 | 19.05 | \ | 23.86 | 19.19 | \ | 19.51 | 13.02 | \ | 35.39 | 6.31 | \ | 20.58 | 19.04 |
| DB_TransE | 14.22 | 14.45 | 14.46 | 20.66 | 23.61 | 20.71 | 13.20 | 14.71 | 13.23 | 6.34 | 57.27 | 6.43 | 20.00 | 21.55 | 17.73 |
| DB_TransH | 13.88 | 12.81 | 14.28 | 20.71 | 23.59 | 20.72 | 13.04 | 14.19 | 13.03 | 6.35 | 57.27 | 6.47 | 19.88 | 21.54 | 16.66 |
| DB_TransR | 14.50 | 13.24 | 14.57 | 20.10 | 23.37 | 20.04 | 13.87 | 15.74 | 13.93 | 6.34 | 57.31 | 6.37 | 20.45 | 21.55 | 17.18 |
| DB2vec SG 200w 200v 4d | 14.26 | 14.02 | 15.15 | 16.82 | 18.52 | 16.56 | 11.57 | 12.38 | 11.56 | 6.26 | 57.20 | 6.45 | 19.68 | 20.85 | 18.50 |
| DB2vec CBOW 200w 200v 4d | 14.13 | 13.33 | 15.46 | 18.66 | 20.61 | 18.39 | 12.32 | 13.87 | 12.32 | 6.35 | 56.85 | 6.37 | 19.67 | 20.84 | 18.72 |
| DB2vec CBOW 500w 200v 4d | 14.37 | 12.55 | 14.33 | 15.90 | 17.46 | 15.89 | 11.79 | 12.45 | 11.59 | 12.13 | 45.76 | 12.00 | 18.32 | 26.19 | 17.43 |
| DB2vec CBOW 500w 500v 4d | 14.99 | 12.46 | 14.66 | 15.90 | 17.45 | 15.73 | 11.49 | 12.60 | 11.48 | 12.44 | 45.67 | 12.30 | 18.23 | 26.27 | 17.62 |
| DB2vec SG 500w 200v 4d | 13.38 | 12.54 | 15.13 | 15.81 | 17.07 | 15.84 | 11.30 | 12.36 | 11.42 | 12.13 | 45.72 | 12.10 | 17.63 | 26.13 | 17.85 |
| DB2vec SG 500w 500v 4d | 14.73 | 13.25 | 16.80 | 15.66 | 17.14 | 15.67 | 11.20 | 12.11 | 11.28 | 12.09 | 45.76 | 11.93 | 18.23 | 26.09 | 17.74 |
| DB2vec CBOW 500w 200v 8d | 16.17 | 17.14 | 17.56 | 21.55 | 23.75 | 21.46 | 13.35 | 15.41 | 13.43 | 6.47 | 55.76 | 6.47 | 24.17 | 26.48 | 22.61 |
| DB2vec CBOW 500w 500v 8d | 18.13 | 17.19 | 18.50 | 20.77 | 23.67 | 20.69 | 13.20 | 15.14 | 13.25 | 6.54 | 55.33 | 6.55 | 21.16 | 25.90 | 20.33 |
| DB2vec SG 500w 200v 8d | 12.85 | 14.95 | 12.92 | 15.15 | 17.13 | **15.12** | 10.90 | 11.43 | 10.90 | 6.22 | 56.95 | 6.25 | 18.66 | 21.20 | 18.57 |
| DB2vec SG 500w 500v 8d | 11.92 | 12.67 | **10.19** | 15.45 | 17.80 | 15.50 | **10.89** | 11.72 | 10.97 | 6.26 | 56.95 | 6.29 | 18.35 | 21.04 | **16.61** |
| WD2vec CBOW 200w 200v 4d | 20.15 | 17.52 | 20.02 | 23.54 | 25.90 | 23.39 | 14.73 | 16.12 | 14.55 | 16.80 | 42.61 | 6.60 | 27.48 | 22.60 | 21.77 |
| WD2vec CBOW 200w 500v 4d | 23.76 | 18.33 | 20.39 | 24.14 | 22.18 | 24.56 | 14.09 | 16.09 | 14.00 | 13.08 | 42.89 | 6.08 | 50.23 | 21.92 | 26.66 |
| WD2vec SG 200w 200v 4d | 20.47 | 18.69 | 20.72 | 19.72 | 21.44 | 19.10 | 13.51 | 13.91 | 13.67 | 6.86 | 42.82 | 6.52 | 23.69 | 21.59 | 20.49 |
| WD2vec SG 200w 500v 4d | 22.25 | 19.41 | 19.23 | 25.99 | 21.26 | 19.19 | 13.23 | 14.96 | 13.25 | 8.27 | 42.84 | **6.05** | 21.98 | 21.73 | 21.58 |

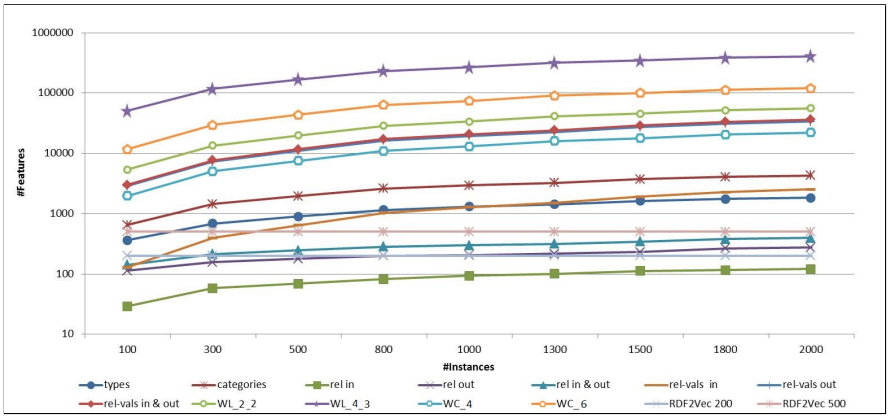



Figure 8.3: Features increase rate per strategy (log scale).

## 8.6 Features Increase Rate

Finally, we conduct a scalability experiment, where we examine how the number of instances affects the number of generated features by each feature generation strategy. For this purpose we use the *Metacritic Movies* dataset. We start with a random sample of 100 instances, and in each next step we add 200 (or 300) unused instances, until the complete dataset is used, i.e., $2,000$ instances. The number of generated features for each sub-sample of the dataset using each of the feature generation strategies is shown in Figure 8.3.

From the chart, we can observe that the number of generated features sharply increases when adding more samples in the datasets, especially for the strategies based on graph substructures.

In contrast, the number of features remains the same when using the RDF2Vec approach, as it is fixed to 200 or 500, respectively, independently of the number of samples in the data. Thus, by design, it scales to larger datasets without increasing the dimensionality of the dataset.

## 8.7 Conclusion and Outlook

In this chapter, we have presented RDF2Vec, an approach for learning latent numerical representations of entities in RDF graphs. In this approach, we first convert the RDF graphs in a set of sequences using two strategies, Weisfeiler-Lehman Subtree RDF Graph Kernels and graph walks, which are then used to build neural language models. The evaluation shows that such entity representations could be used in three different tasks. In each of those tasks, they were capable of outperforming standard feature generation approaches, i.e., approaches that turn (subsets of) RDF graphs into propositional models.

We have explored different variants of building embedding models. While there is no universally best performing approach, we can observe some trends.

With respect to the first step of the transformation, i.e., the construction of sequences, kernel transformations lead to better results than (random) walks. However, they do not scale well to large-scale knowledge graphs, such as DBpedia or Wikidata. With respect to the second step, i.e., the actual computation of the embeddings, we have observed that Skip-Gram (SG) models in most cases outperform Continuous-Bag-of-Words (CBOW) models. The other characteristics of the models (e.g., the dimensionality of the embedding space) show less clear trends towards an optimal setting.

While constructing a vector space embedding for a large-scale knowledge graph, such as DBpedia or Wikidata, can be computationally expensive, we have shown that this step has to be taken only once, as the embeddings can be reused on various tasks. This is particularly interesting for such cross-domain knowledge graphs, which can be used in a variety of scenarios and applications.

For the moment, we have defined some constraints for the construction of the embeddings. We do not use literal values, and we do not particularly distinguish between the schema and the data level of a graph. The former constraint has some limitations, e.g., when it comes to the tasks of determining entity similarity: for example, the similarity of two movies in terms of release date and budget or the similarity of two cities in terms of area and population is currently not captured by the models. Schema level and data level similarity are currently implicitly interwoven, but in particular for knowledge graphs with richer schemas (e.g., YAGO with its type hierarchy of several hundred thousand types), distinguishing embeddings of the schema and data level might become beneficial.

Apart from using vector space embeddings when exploiting LOD data sources, they may also become an interesting technique for improving those sources as such, for example knowledge base completion [167]. Among others, the proposed approach could also be used for link prediction, entity typing, or error detection in knowledge graphs [223], as shown in [181, 195]. Similarly to the entity and document modeling, the approach can be extended for entity summarization, which is also an important task when consuming and visualizing large quantities of data [42].

Summarizing, we have shown that it is possible to adapt the technique of word embeddings to RDF graphs, and that those embeddings lead to compact vector representations of entities. We have shown that those vector representations help building approaches which outperform many state of the art tools. Furthermore, the proposed vector space representations are *universal* in the sense that they are not task specific, i.e., a vector space embedding for a general graph like DBpedia or Wikidata can be built once and reused for several tasks.

# Chapter 9

# Biased Graph Walks for RDF Graph Embeddings

In the previous chapter, we introduced *RDF2Vec*, a generic method for embedding entities in knowledge graphs into lower-dimensional vector spaces. The approach adapts neural language modeling techniques, specifically word2vec, which takes sequences of words to embed words into vector spaces [178, 177]. We have shown that it is possible to compute and reuse such embeddings for large-scale knowledge graphs.

For adapting word2vec for knowledge graphs, the first step is to extract meaningful sequences of entities from a knowledge graph, which capture the surrounding knowledge of each entity. Our results have shown that *random walks* are a feasible and, in contrast to other techniques such as kernels, also a well scalable approach for extracting sequences.

In this chapter, we examine methods to direct the random walks in more meaningful ways, i.e., being able to capture more important information about each entity in the graph. We test a dozen weighting schemes which influence the walks and, thus, the resulting sequences. The experiments show that the choice of weights has a crucial influence on the utility of the resulting embeddings.

**The work presented in this chapter has been published before as: "Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, Heiko Paulheim: *Biased Graph Walks for RDF Graph Embeddings.* Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics, Amantea, Italy, June, 2017." [44].**
**The implementation of the biased walks was carried by Michael Cochez, from the Fraunhofer Institute for Applied Information Technology FIT, 53754 Sankt Augustin, Germany.**

## 9.1 Approach

In our approach, we adapt neural language models for RDF graph embeddings. Such approaches take advantage of the word order in text documents, explicitly modeling the assumption that closer words in the word sequences are statistically more dependent. In the case of RDF graphs, we consider entities and relations between entities instead of word sequences. Thus, in order to apply such approaches on RDF graph data, we first have to transform the graph data into sequences of entities, which can be considered as sentences. Using those sequences, we can train the same neural language models to represent each entity in the RDF graph as a vector of numerical values in a latent feature space.

We use graph walks for converting graphs into a set of sequences of entities. An example of an entity sequence extracted using graph walks from DBpedia would be: $dbr : Trent\_Reznor \rightarrow dbo : associatedBand \rightarrow dbr : Nine\_Inch\_Nails \rightarrow dbo : genre \rightarrow dbr : Industrial\_Rock$. To perform these walks on RDF graphs, we represent the graph as a set of vertices (the entities in the RDF graph) and a set of directed edges (the relations between the entities).

The objective of the walk functions is for each vertex $v \in V$ to generate a set of sequences $S_v$, where the first token of each sequence $s \in S_v$ is the vertex $v$ followed by a sequence of tokens, which might be the labels of edges, vertices, or any substructure extracted from the RDF graph, in an order that reflects the relations between the vertex $v$ and the rest of the tokens, as well as among those tokens.

What we want to achieve is a biasing of these walks to make them more meaningful, i.e., being able to capture the most important information about the observed entities. Therefore, we augment the edges to not only have a label, but also a weight. We apply twelve different strategies for assigning these weights to the edges of the graph. These weights will then in turn bias the random walks on the graph. In particular, when a walk arrives in a vertex $v$ with out edges $v_{o1}, \ldots v_{od}$, then the walk will follow edge $v_{ol}$ with a probability computed by

$$\Pr[\text{follow edge } v_{ol}] = \frac{weight(v_{ol})}{\sum_{i=1}^{d} weight(v_{oi})}$$

In other words, the normalized edge weights are directly interpreted as the probability to follow a particular edge.

To obtain these edge weights, we make use of different statistics computed on the RDF data. The statistics computed are the following:

**Predicate Frequency** for each predicate in the dataset, we count the number of times the predicate occurs (only occurrences as a predicate are counted).

**Object Frequency** for each resource in the dataset, we count the number of times it occurs as the object of a triple.

**Predicate-Object frequency** for each pair of a predicate and an object in the dataset, we count the number of times there is a statement with this predicate and object.

Besides these statistics, we also use PageRank [29] computed for the entities in the knowledge graph [296]. This PageRank is computed based on links between the Wikipedia articles representing the respective entities. When using the PageRank computed for DBpedia, not each node has a value assigned, as only entities which have a corresponding Wikipedia page are accounted for in the PageRank computation. Examples of nodes which do not have a PageRank include DBpedia types or categories, like `http://dbpedia.org/ontology/Place` and `http://dbpedia.org/resource/Category:Central_Europe`. Therefore, we assigned a fixed PageRank to all nodes which are not entities. We chose a value of 0.2, which is roughly the median PageRank [311], in the non-normalized page rank values we used.

Note that there are essentially two types of metrics, those assigned to nodes, and those assigned to edges. The predicate frequency and predicate-object frequency, as well as the inverses of these, can be directly used as weights for edges. Therefore, we call these weighting methods *edge-centric*. In the case of predicate frequency each predicate edge with that label is assigned the weight in question. In the case of predicate-object frequency, each predicate edge which ends in a given object gets assigned the predicate-object frequency. When computing the inverse metrics, not the absolute frequency is assigned, but its multiplicative inverse.

In contrast, the object frequency, and also the used PageRank metric, assign a numeric score to each node in the graph. Therefore, we call weighting approaches based on them *node-centric*. To obtain a weight for the edges, we either *push* the number down or *split* the number down to all in edges. By pushing down, we mean that the number assigned to a node is used as the weight of all in edges. By splitting down, we mean that the weight is divided by the number of in edges and then assigned to all edges. Then, these weights can be normalized as described above. If *split* is not mentioned explicitly in node centric weighting strategies, then it is a push down strategy.

In total, we inspected twelve different approaches for weighting edges using the metrics defined above.

Note that uniform weights are equivalent to using object frequency with splitting the weights. To see why this holds true, we have to follow the steps which will be taken. First, each node gets assigned the amount of times it is used as an object. This number is equal to the number of in edges to the node. Then, this number is split over the in edges, i.e., each in edge gets assigned the number 1. Finally, this weight is normalized, assigning to each out link a uniform weight. Hence, this strategy would result in the same walks as using unbiased random walks over the graph.

So, even if we add unbiased random walks to the list of weighting strategies, we retain 12 unique ones, each with their own characteristics. These strategies are:

Uniform approach:

1. *Uniform Weight = Object Frequency Split Weight* – This is the most straight forward approach, also taken by the standard RDF2Vec models. At first glance, it also looks like the most neutral strategy. However, the input graph does not have a regular structure in the sense that some entities have a (much) higher in degree as others and hence they are more likely to be visited. Thus, more strongly connected entities will have a higher influence on the resulting embeddings.

Edge-centric approaches:

2. *Predicate Frequency Weight* – With this strategy, edges with predicates which are commonly used in the dataset are more often followed. The effect of this is that many uncommon predicates are never followed in our experiments and, as a result of that, many entities are also never visited in the walks. On the other hand, there are a few entities which have a very high in degree, and which thus attract a lot of walks towards them.

3. *Inverse Predicate Frequency Weight* – This strategy has at first sight a similar effect as the previous, but for other nodes. Those predicates which are rare will be followed often. However, predicates follow a long-tail distribution, and there are more predicates which are rare than common, thus, the diversity of predicates occurring in the walks is higher. Moreover, despite having a low probability, also edges with a common predicate are followed once in a while as they occur so often in the dataset.

4. *Predicate-Object Frequency Weight* – This is similar to the Predicate Frequency Weight, but differentiates between the objects as well. If we have for example an outgoing link with label `rdf:type` with object `owl:Thing`, then this link will be followed more often than, e.g., the same predicate with object `dbpedia:AdministrativeRegion`.

5. *Inverse Predicate-Object Frequency Weight* – The inverse of the previous, with similar features to Inverse Predicate Frequency Weight.

Node-centric object freq. approaches (See also strategy 1):

6. *Object Frequency Weight* – This weighting does essentially ignore the predicate altogether and just ensures that entities which have a high in degree get visited even more often.

7. *Inverse Object Frequency Weight* – This approach also ignores the predicate, but makes the probability for nodes to be visited more equally distributed.

8. *Inverse Object Frequency Split Weight* – The general statistics for these walks look surprisingly similar to the non inverted strategy.

Node-centric PageRank approaches:

9. *PageRank Weight* – Similar to Object Frequency Weight, this strategy makes some nodes more important and hence there will be resources which are more frequent in the walks as others.

10. *Inverse PageRank Weight* – One would expect that this approach would have a similar effect as Inverse Object Frequency Weight, however, our measurements show that the inversion does not cause more uniform occurrence of entities as strongly as that strategy.

11. *PageRank Split Weight* – Both this approach and the next one are somewhat difficult to predict as they do not only depend on the structure on the graph. Our analysis of the walks show that nodes are fairly uniformly used in these walks.

12. *Inverse PageRank Split Weight* – The generated walks have similar statistics as PageRank Split Weight. The expectation is, however, that in this metric tends to include more unimportant nodes in the walks.

For each set of the twelve sets of sequences created using those metrics, we build one CBOW and one skip-gram model. Hence, we compare a total of 24 different embeddings models.

## 9.2 Evaluation

Similarly as in Chapter 8, we evaluate the different weighting strategies on a number of classification and regression tasks, comparing the results of different feature extraction strategies combined with different learning algorithms.

To build the neural language models, we generate 250 walks per entity with depths of 2,4,6, and 8 for each of the twelve edge weighting strategies. A depth of eight means four hops in the graph, as each hop adds two elements to the sequence (i.e., the predicate and the object). Since, the entity which is the source of the walk is also include in the path, the corresponding path lengths are 3,5,7, and 9. When the walk reaches a "dead end", i.e., a node without any outgoing edges, the walk ends in that node, even if the maximum depth is not reached.

We use the corpora of sequences to build both CBOW and Skip-Gram models with the following parameters: window size = 5; number of iterations = 5; negative sampling for optimization; negative samples = 25; dimensions = 200; with average input vector for CBOW. The parameters are selected based on recommendations from the literature. All the models, as well as the code, are publicly available[1].

[1] `http://data.dws.informatik.uni-mannheim.de/rdf2vec/`

### 9.2.1 Datasets

We evaluate our approach on DBpedia [162]. We use the English version of the 2016-04 DBpedia dataset, which contains $4,678,230$ instances and $1,379$ mapping-based properties. In our evaluation we only consider object properties, and ignore datatype properties and literals.

We use the entity embeddings on five different datasets from different domains, for the tasks of classification and regression [247], used in Chapter 8. Those five datasets are used to provide classification/regression targets for the large RDF datasets (see Table 11.1).

- The *Cities* dataset contains a list of cities and their quality of living, as captured by Mercer[2]. We use the dataset both for regression and classification.
- The *Metacritic Movies* dataset is retrieved from Metacritic.com[3], which contains an average rating of all time reviews for a list of movies [256]. The initial dataset contained around $10,000$ movies, from which we selected $1,000$ movies from the top of the list, and $1,000$ movies from the bottom of the list. We use the dataset both for regression and classification.
- Similarly, the *Metacritic Albums* dataset is retrieved from Metacritic.com[4], which contains an average rating of all time reviews for a list of albums [257].
- The *AAUP* (American Association of University Professors) dataset contains a list of universities, including eight target variables describing the salary of different staff at the universities[5]. We use the average salary as a target variable both for regression and classification, discretizing the target variable into "high", "medium" and "low", using equal frequency binning.
- The *Forbes* dataset contains a list of companies including several features of the companies, which was generated from the Forbes list of leading companies 2015[6]. The target is to predict the company's market value as a regression task. To use it for the task of classification we discretize the target variable into "high", "medium", and "low", using equal frequency binning.

### 9.2.2 Experimental Setup

As in Chapter 8, we compare our approach to several baselines. For generating the data mining features, we use three strategies that take into account the direct relations to other resources in the graph [225], and two strategies for features derived from graph sub-structures [59]:

[2] https://www.imercer.com/content/mobility/quality-of-living-city-rankings.html

[3] http://www.metacritic.com/browse/movies/score/metascore/all

[4] http://www.metacritic.com/browse/albums/score/metascore/all

[5] http://www.amstat.org/publications/jse/jse_data_archive.htm

[6] http://www.forbes.com/global2000/list/

Table 9.1: Classification and regression datasets overview. For each dataset, we depict the number of instances, the machine learning tasks in which the dataset is used (*C* stands for classification, and *R* stands for regression) and the source of the dataset.

| Dataset | #Instances | ML Task | Original Source |
|---|---|---|---|
| Cities | 212 | R/C (c=3) | Mercer |
| Metacritic Albums | 1600 | R/C (c=2) | Metacritic |
| Metacritic Movies | 2000 | R/C (c=2) | Metacritic |
| AAUP | 960 | R/C (c=3) | JSE |
| Forbes | 1585 | R/C (c=3) | Forbes |

- Features derived from specific relations. In the experiments we use the relations *rdf:type* (types), and *dcterms:subject* (categories).
- Features derived from generic relations, i.e., we generate a feature for each incoming (rel in) or outgoing relation (rel out) of an entity, ignoring the value or target entity of the relation.
- Features derived from generic relations-values, i.e, we generate feature for each incoming (rel-vals in) or outgoing relation (rel-vals out) of an entity including the value of the relation.
- Kernels that count substructures in the RDF graph around the instance node. These substructures are explicitly generated and represented as sparse feature vectors.
  - The Weisfeiler-Lehman (WL) graph kernel for RDF [59] counts full subtrees in the subgraph around the instance node. This kernel has two parameters, the subgraph depth $d$ and the number of iterations $h$ (which determines the depth of the subtrees). We use two pairs of settings, $d = 1, h = 2$ and $d = 2, h = 3$.
  - The Intersection Tree Path kernel for RDF [59] counts the walks in the subtree that spans from the instance node. Only the walks that go through the instance node are considered. We will therefore refer to it as the root Walk Count (WC) kernel. The root WC kernel has one parameter: the length of the paths $l$, for which we test 2 and 3.

Furthermore, we compare the results to the state-of-the art graph embeddings approaches: TransE, TransH and TransR. We use an existing implementation and build models on the DBpedia data with the default parameters.[7]

We perform two learning tasks, i.e., classification and regression. For classification tasks, we use Naive Bayes, k-Nearest Neighbors (k=3), C4.5 decision tree, and Support Vector Machines. For the SVM classifier we optimize the parameter *C* in the range $\{10^{-3}, 10^{-2}, 0.1, 1, 10, 10^2, 10^3\}$. For regression, we use Linear

[7] https://github.com/thunlp/KB2E/

Table 9.2: Classification average rank results. The best ranked results for each method are marked in bold. The learning models for which the strategies were shown to have significant difference based on the Friedman test with $\alpha < 0.05$ are marked with *. The single values marked with $*$ mean that are significantly worse than the best strategy at significance level $q = 0.05$

| Method | | NB* | KNN* | SVM | C4.5 |
|---|---|---|---|---|---|
| Uniform Weight | CBOW | 14.4 | 9.7 | 12.8 | 9.4 |
| | SG | 6.4 | **3.3** | 10.0 | **6.6** |
| Edge-centric approaches | | | | | |
| Predicate Frequency Weight | CBOW | 14.0 | 11.3 | 12.6 | 14.0 |
| | SG | 11.6 | 11.1 | 10.4 | 12.8 |
| Inverse Predicate Frequency Weight | CBOW | 24.6* | 25.6* | 22.5 | 19.8 |
| | SG | 23.0 | 19.4 | 15.8 | 18.2 |
| Predicate Object Frequency Weight | CBOW | 20.5 | 20.9 | 17.9 | 20.8 |
| | SG | 20.4 | 20.3 | 16.7 | 20.6 |
| Inverse Predicate Object Frequency Weight | CBOW | 19.0 | 16.8 | 15.3 | 15.4 |
| | SG | 17.2 | 15.6 | 10.6 | 12.2 |
| Node-centric object freq. approaches | | | | | |
| Object Frequency Weight | CBOW | 19.1 | 20.2 | 17.9 | 21.0 |
| | SG | 17.8 | 14.6 | 14.0 | 15.8 |
| Inverse Object Frequency Weight | CBOW | 7.0 | 10.6 | 10.2 | 7.6 |
| | SG | 19.6 | 19.4 | 15.7 | 21.0 |
| Inverse Object Frequency Split Weight | CBOW | 18.8 | 16.7 | 16.0 | 13.4 |
| | SG | 7.4 | 10.9 | 13.1 | 14.2 |
| Node-centric PageRank approaches | | | | | |
| PageRank Weight | CBOW | 25.2* | 22.6 | 20.9 | 19.0 |
| | SG | 14.2 | 9.8 | 9.8 | 13.0 |
| Inverse PageRank Weight | CBOW | 8.2 | 14.8 | 12.4 | 10.6 |
| | SG | 4.8 | 10.0 | 9.8 | 9.0 |
| PageRank Split Weight | CBOW | 23.4 | 10.9 | 17.0 | 15.2 |
| | SG | **4.4** | 4.7 | **6.7** | 8.4 |
| Inverse PageRank Split Weight | CBOW | 13.4 | 11.3 | 17.9 | 15.6 |
| | SG | 7.4 | 8.9 | 11.6 | 10.6 |
| Baseline and related approaches | | | | | |
| Best Baseline | | 12.0 | 15.0 | 19.0 | 7.8 |
| TransE | | 10.0 | 16.7 | 16.8 | 16.6 |
| TransH | | 9.8 | 15.8 | 16.3 | 17.2 |
| TransR | | 12.4 | 19.1 | 16.3 | 20.2 |

Table 9.3: Regression average rank results. The best ranked results for each method are marked in bold. The learning models for which the strategies were shown to have significant difference based on the Friedman test with $\alpha < 0.05$ are marked with *. The single values marked with $*$ mean that are significantly worse than the best strategy at significance level $q = 0.05$

| Method | | LR* | KNN | M5 |
|---|---|---|---|---|
| Uniform Weight | CBOW | 8.0 | 7.4 | 9.0 |
| | SG | **4.4** | 7.6 | 8.8 |
| Edge-centric approaches | | | | |
| Predicate Frequency Weight | CBOW | 10.8 | 13.4 | 10.8 |
| | SG | 15.0 | 11.6 | 16.4 |
| Inverse Predicate Frequency Weight | CBOW | 22.0 | 16.8 | 21.6 |
| | SG | 13.0 | 15.4 | 17.2 |
| Predicate Object Frequency Weight | CBOW | 24.6* | 22.4 | 24.2 |
| | SG | 24.8* | 23.6 | 24.8 |
| Inverse Predicate Object Frequency Weight | CBOW | 12.6 | 14.0 | 13.4 |
| | SG | 6.2 | 10.6 | 8.2 |
| Node-centric object freq. approaches | | | | |
| Object Frequency Weight | CBOW | 22.8 | 22.2 | 21.6 |
| | SG | 10.8 | 15.0 | 14.6 |
| Inverse Object Frequency Weight | CBOW | 6.8 | 10.0 | 9.4 |
| | SG | 26.0* | 22.8 | 23.8 |
| Inverse Object Frequency Split Weight | CBOW | 21.0 | 20.2 | 19.0 |
| | SG | 13.2 | 15.6 | 13.2 |
| Node-centric PageRank approaches | | | | |
| PageRank Weight | CBOW | 25.8* | 18.0 | 25.6 |
| | SG | 7.0 | 15.4 | 7.8 |
| Inverse PageRank Weight | CBOW | 11.4 | 8.8 | 13.0 |
| | SG | 7.4 | **6.8** | **6.2** |
| PageRank Split Weight | CBOW | 17.6 | 12.2 | 17.8 |
| | SG | 8.6 | 10.2 | 8.4 |
| Inverse PageRank Split Weight | CBOW | 17.6 | 18.2 | 17.8 |
| | SG | 9.4 | 11.2 | 7.2 |
| Baseline and related approaches | | | | |
| Best Baseline | | 17.4 | 9.6 | 9.6 |
| TransE | | 12.8 | 16.7 | 13.0 |
| TransH | | 12.8 | 14.1 | 12.4 |
| TransR | | 16.2 | 16.2 | 11.2 |

Regression, M5Rules, and k-Nearest Neighbors (k=3). The results are calculated using stratified 10-fold cross validation.

The strategies for creating propositional features from Linked Open Data are implemented in the RapidMiner LOD extension[8] [229, 246]. The experiments, including the feature generation and the evaluation, were performed using the RapidMiner data analytics platform.[9] The RapidMiner processes and the complete results can be found online.[10]

For comparing the approaches, we follow the approach introduced by Demšar [60]. The approach proposes to first rank the strategies for each dataset in isolation, and then to compute a significance level for the difference of ranks using a Friedman test. While the Friedman test only determines whether there is a significant difference between *any* of the compared approaches, pairwise significance levels are computed with a post-hoc Nemenyi test [194]. The results of the post-hoc test allows for concluding if one approach significantly outperforms another one. For the Friedman test we select a significance level of $\alpha = 0.10$, and for the post-hoc Nemenyi test we use critical values $q = 0.05$. We carry out the test on each learning method separately.

### 9.2.3 Results

The results for the task of classification on the five different datasets using four different learning methods are given in Table 9.2. For each of the datasets and for each learning method, we select the best performing results of all the baselines, and report it under *Best baseline*. Using the Friedman test, the null hypothesis was rejected for the performances of the strategies when using Naive Bayes and KNN, meaning there is a significant performance difference between the strategies.

The results for the task of regression on the five different datasets using four different learning methods are given in Table 9.3. Using the Friedman test, the null hypothesis was rejected for the performances of the strategies when using Linear Regression, meaning there is a significant performance difference between the strategies.

From the results for both tasks we can conclude that the RDF2Vec approach outperforms the baseline approaches and also outperforms the state-of-the art graph embeddings models. Furthermore, *Inverse PageRank Weight* and *PageRank Split Weight* strategies perform well for different learning methods. Overall, the skip-gram models outperform the corresponding CBOW models for most of the strategies. Unexpectedly, the *Uniform Weight* strategy also yields competitive results.

However, for the variety of tasks at hand, there is no universal approach, i.e., embedding model and a machine learning method, that consistently outperforms

[8] `http://dws.informatik.uni-mannheim.de/en/research/rapidminer-lod-extension`

[9] `https://rapidminer.com/`

[10] `http://data.dws.informatik.uni-mannheim.de/rmlod/LOD_ML_Datasets/`

the others.

## 9.3 Conclusion and Outlook

Vector space embeddings for RDF graphs have been proven a high utility and powerful approach for transforming RDF data and knowledge graphs to propositional forms. The RDF2Vec approach, first introduced in [254], leverages random walks for transforming RDF graphs to token sequences, which is a necessary approach to be able to apply standard vector space embeddings techniques like CBOW and Skip-Gram.

In this chapter, we have examined the influence of edge weights and transition probabilities to guide the walks, i.e., to make them less uniformly random. We have shown that introducing biases to the walks can lead to significant improvements. In particular, the PageRank split and the inverse PageRank weighting schemes provide good results.

So far, we have based our evaluations on machine learning tasks. For future work, we will also study the effect on other tasks in which knowledge graph embeddings have been applied successfully, such as content-based recommender systems [66], as well as link prediction, type prediction, or graph completion and error detection in knowledge graphs [223], as discussed in [181, 196].

In our experiments, we have also experienced that there is not a one-size-fits-all solution for the weighting schemes. Although there are some trends that can be observed, the performance of the weighting schemes is hard to predict in individual cases. Among others, we assume that one crucial factor is the popularity of entities: for example, for very popular entities, the PageRank heuristic is assumed to work well, because it extracts more sequences containing popular entities, while for tail entities, the inverse PageRank heuristic will lead to better results. Future evaluations should examine those effects more deeply.

This work has been continued in [45], where we present an approach that exploits global patterns for creating vector space embeddings, inspired by the Global Vectors (GloVe) [232] approach for learning vector space embeddings for words from a text corpus. We show that using the GloVe approach on the same data as the older RDF2Vec approach does not improve the created embeddings. However, this approach is able to incorporate larger portions of the graph, without substantially increasing the computational time, leading to comparable results.

# Part III

# Applications of Semantic Web Knowledge Graphs

# Chapter 10

# Analyzing Statistics with Background Knowledge from Semantic Web Knowledge Graphs

Statistical datasets are widely spread and published on the Web. However, many users' information need is not the mere consumption of statistical data as such, but the search for patterns and explanations. As shown in the previous chapter, information from the Linked Open Data cloud can serve as background knowledge for interpreting statistical data, as it covers various domains, ranging from general purpose datasets to government and life science data [267].

In this chapter, we present the Web-based tool *ViCoMap*[1], which allows automatic correlation analysis and visualizing statistical data on maps using Semantic Web knowledge graphs. The tool automatically enriches statistical datasets, imported from Semantic Web knowledge graphs, RDF datacubes, or local datasets, with information from Semantic Web knowledge graphs, and uses that background knowledge as a means to create possible interpretations as well as advanced map visualization of the statistical datasets. To visualize geospatial entities on a map, we use GADM[2], a LOD database of polygon shapes of the world's administrative areas.

So far, many tools for visualization of LOD and statistical data have been developed [230]. In particular, for RDF data cubes[3] exposing statistical data, different browsers have been developed, such as *CubeViz*[4] or *Payola*[5] [150]. The *CODE*

[1] The tool is available at `http://vicomap.informatik.uni-mannheim.de/`
[2] `http://gadm.geovocab.org/`
[3] `http://www.w3.org/TR/vocab-data-cube/`
[4] `http://aksw.org/Projects/CubeViz.html`
[5] `http://live.payola.cz/`

*Visualisation Wizard*[6] [191] also features different chart and map based visualizations. Tialhun et al. [303] have developed a LOD-based visualization system for healthcare data. The system visualizes different healthcare indicators per country on a map, and is able to perform correlation analysis between selected indicators, which can later be visualized as a chart.

The direct predecessor of *ViCoMap* is *Explain-a-LOD* [219], which is one of the first approaches for automatically generating hypothesis for explaining statistics using LOD. The tool enhances statistical datasets with background information from DBpedia[7], and uses correlation analysis and rule learning for producing hypothesis which are presented to the user.

*ViCoMap* combines map-based visualizations on the one hand side, and mining for correlations using background knowledge from LOD on the other. As such, it opens new ways of interpreting statistical data.

**The work presented in this chapter has been published before as: "Petar Ristoski, Heiko Paulheim: *Visual Analysis of Statistical Data on Maps Using Linked Open Data.* Revised Selected Papers of the 12th European Semantic Web Conference, Portoroz, Slovenia, May 2015 " [250].**

## 10.1 The ViCoMap Tool

The architecture of the ViCoMap tool consists of three main components, as shown in Fig. 10.1. The base component of the tool is the *RapidMiner Linked Open Data extension* [246]. The extension hooks into the powerful data mining platform *RapidMiner*[8], and offers operators for accessing LOD in RapidMiner, allowing for using it in sophisticated data analysis workflows using data from LOD. The extension allows for autonomously exploring the Web of Data by following links, thereby discovering relevant datasets on the fly, as well as for integrating redundant data found in different datasets, and wraps additional services such as *DBpedia Lookup*[9] and *DBpedia Spotlight*[10].

All processes built within the RapidMiner platform can be exposed as Web services through the RapidMiner Server, which can be consumed in a user Web application. We use such a setup to integrate the functionalities of the RapidMiner LOD extension, as well as the functionalities of RapidMiner built-in operators, in the ViCoMap Web application. The ViCoMap Web application offers three main functionalities to the end-user: *Data Import*, *Correlation Analysis*, and *Visualization on Maps*.

[6] http://code.know-center.tugraz.at/search
[7] http://dbpedia.org/
[8] http://www.rapidminer.com
[9] http://lookup.dbpedia.org
[10] http://lookup.dbpedia.org

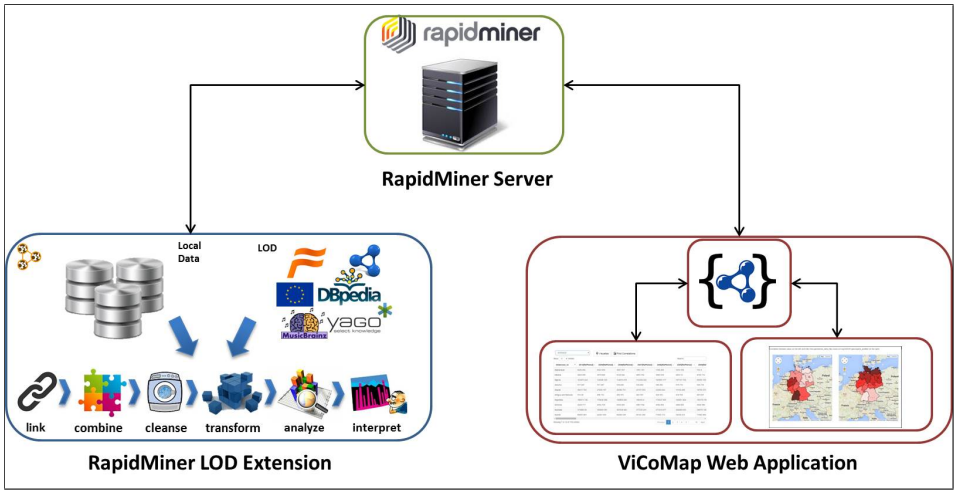



Figure 10.1: ViCoMap Architecture

### 10.1.1 Data Import

There are three options to import data, i.e.,import a dataset published using the RDF Data Cube vocabulary, import data from a SPARQL endpoint, and import data from a local file.

**Import RDF Data Cubes:** To import a dataset published using the RDF Data Cube vocabulary, the user first needs to select the data publisher source, and a dataset that will be explored. Currently, we provide a static list of most used RDF Data Cube publishers, like WorldBank[11]. After selecting a data cube and the dimensions to be analyzed, the dataset is loaded into RapidMiner by the LOD extension.

**SPARQL Data Import:** To import data from a SPARQL endpoint, the user first needs to select a SPARQL endpoint and to provide a SPARQL query. The tool offers a SPARQL query builder assistant, which helps the user formulate queries such as *Select the number of universities per federal state in Germany*, by selecting a set of spatial entities (states in Germany) and a subject entity (universities).

**Local Dataset Import:** The user can import data from a local CSV file.

### 10.1.2 Correlation Analysis

Once the data is loaded, the user can select a column that will be used for correlation analysis. The user can choose the LOD sources that will be explored to find interesting factors that correlate with the target value at hand. To link the data at

[11] `http://worldbank.270a.info`

hand to remote LOD datasets, the tool exploits existing `owl:sameAs` links, and it automatically creates additional links, e.g., via *DBpedia Lookup* for non-linked datasets, such as CSV files. From the data retrieved from the additional LOD sources, a simple correlation analysis is performed to find simple correlations of the generated features and the target value under examination. The discovered correlations are sorted by confidence and presented to the user.

#### Visualization on Maps

After the correlation analysis is completed, the user can visualize any correlation on a map, using the Google Maps API[12] and displaying two maps for the correlated values side by side. The shape data of the geographical entities is retrieved from GADM. DBpedia provides external links to the GADM dataset, which were created using different heuristics based on the label and coordinates of geographical entities [251]. DBpedia 2014 contains $65,616$ links to the GADM dataset, for entities on different administration level, e.g., municipalities, regions, states, departments, countries, etc. Such links allow us to visualize spatial entities on any administrative level.

## 10.2 Use Case: Number of Universities per State in Germany

In this use case, we analyze which factors correlate with the number of universities per state in Germany[13]. To import the initial data we use the query builder assistant from the SPARQL data import tab (Figure 10.2a). After executing the query, the data is presented in a table with two columns, i.e., the DBpedia URI for each state, and the number of universities per state (Figure 10.2b). By pressing the button *Find Correlations*, we can select the LOD sources that will be included in the correlation analysis (Figure 10.2c). Next, the discovered correlations are presented in a new table with two columns, i.e., a column with the factor label, and the a column with the correlation confidence (Figure 10.2d).

We can see that in this case, as shown in Fig. 10.3, the highest positive correlation is the RnD expenses of the states (+0.84), which is retrieved from Eurostat. The highest negative correlation is the latitude of the states (-0.73), which is retrieved from GeoNames, which reflects the north-south gradient of the wealth distribution in Germany.[14]

[12] `https://developers.google.com/maps/`

[13] States of Germany: `http://en.wikipedia.org/wiki/States_of_Germany`

[14] `http://www.bundesbank.de/Redaktion/EN/Topics/2013/2013_07_10_to_save_or_not_to_save_private_wealth_in_germany.html`

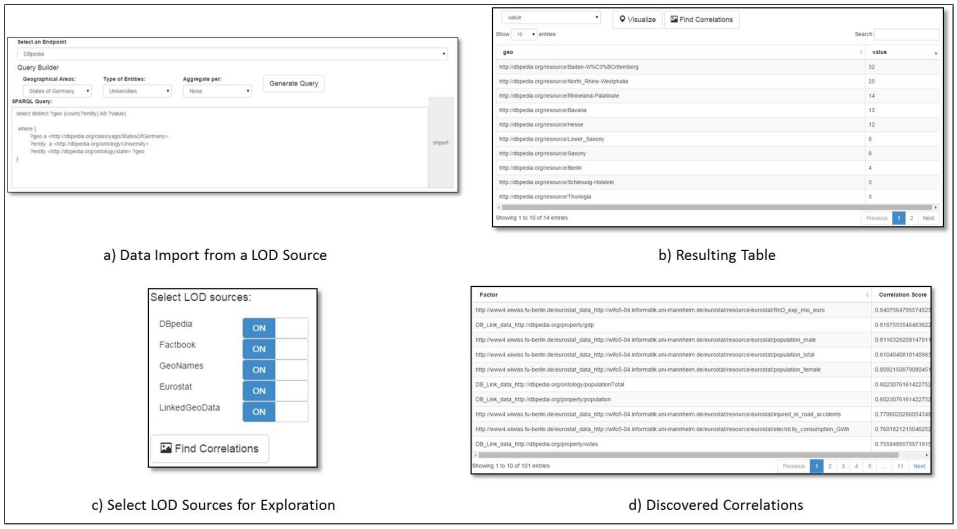
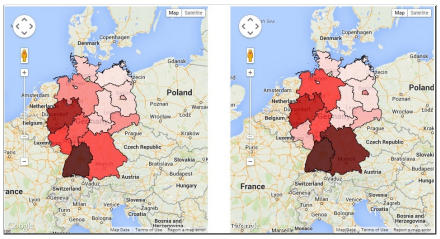
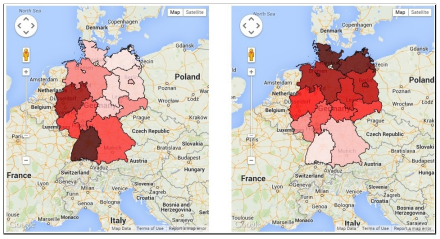



Figure 10.2: German States Use Case Workflow



(a) Positive correlation between #universities (left) and RnD expenses (right) per state



(b) Negative correlation between #universities (left) and latitude (right) per state

Figure 10.3: Correlations visualized on a map using GADM geographical shape data

## 10.3 Conclusion and Outlook

In this chapter, we have introduced the web-based ViCoMap tool, which allows the users to analyze statistical data, and visualize it on maps using external knowledge from Semantic Web knowledge graphs. While at the moment, we use only literal data properties and types for finding correlations, we aim at a more intelligent exploration of the feature space, e.g., by automatically finding meaningful aggregations of different measures.

# Chapter 11

# Semantic Web enabled Recommender Systems

Recommender systems are systems that provide a suggestion of items to a user, based on the user's profile and/or previous behavior. They are used, e.g., for music recommendation in streaming services, in online shopping sites, or on news portals and aggregators. The two major types of recommender systems are *collaborative filtering* and *content-based* recommender systems. The former exploit similarity among *users*, i.e., they recommend items that have been consumed and/or ranked high by users that have similar interests as the user for which the recommendation is made. The latter exploit similarities among *items*, e.g., recommending music of the same genre or news articles on the same topic. Combinations of those approaches, known as *hybrid* approaches, have also been widely studied [33].

In particular for content-based recommender systems, Semantic Web knowledge graphs have been shown to be a valuable source of background knowledge. Despite data from various domains being published as Linked Open Data [267], particularly cross-domain sources such as DBpedia [162] are primarily used in recommender systems. Given that the items to be recommended are linked to a LOD dataset, information from LOD can be exploited to determine which items are considered to be similar to the ones that the user has consumed in the past. For example, DBpedia holds information about genres of books and music recordings, which can be exploited in recommendation systems [116]. Most often, selected data is extracted from DBpedia and transformed into a propositional form, i.e., each graph node is represented by a flat vector of binary and/or numeric features. However, DBpedia contains more information than expressed in those propositional forms. In particular, semantic *paths* between entities are a good candidate for building cross-domain recommender systems.

In this chapter, we present three approaches for exploiting Semantic Web knowledge graphs for building recommender systems. The first approach is based on graph metrics, the second approach is based on a hybrid approach using flat features extracted from Semantic Web knowledge graphs, and the third approch is

based on Semantic Web knowledge graph embeddings.

**The work presented in this chapter has been published before as: "Petar Ristoski, Michael Schuhmacher, Heiko Paulheim: *Using Graph Metrics for Linked Open Data Enabled Recommender Systems.* Proceedings of the 16th International Conference on Electronic Commerce and Web Technologies, Valencia, Spain, September, 2015." [259], "Petar Ristoski, Eneldo Loza Mencía, Heiko Paulheim: *A hybrid multi-strategy recommender system using linked open data.* Semantic Web Evaluation Challenge, Crete, Greece, May, 2014" [249], "Jessica Rosati, Petar Ristoski, Tommaso Di Noia, Renato de Leone, Heiko Paulheim: *RDF graph embeddings for content-based recommender systems.* Proceedings of the 3rd Workshop on New Trends in Content-based Recommender Systems, in conjunction with the 10th ACM Conference on Recommender Systems Boston, MA, USA, September 16, 2016."' [261], "Petar Ristoski, Jessica Rosati, Tommaso Di Noia, Renato de Leone, Heiko Paulheim: *RDF2Vec: RDF Graph Embeddings and Their Applications.* The Semantic Web Journal." [258]**

**The evaluation of the recommender system presented in section 11.4, was carried by Jessica Rosati, from the Polytechnic University of Bari – Via Orabona, 4 – 70125 Bari, Italy.**

## 11.1 Related Work

It has been shown that LOD can improve recommender systems towards a better understanding and representation of user preferences, item features, and contextual signs they deal with. LOD has been used in content-based, collaborative, and hybrid techniques, in various recommendation tasks, i.e., rating prediction, Top-N recommendations and diversity in content-based recommendations. An overview of cross-domain recommender systems is given in [37].

Among the earliest such efforts is *dbrec* [216], which uses DBpedia as knowledge base to build a content-based music recommender system. The recommendations are based on measure (named Linked Data Semantic Distance) which computes the distance between two items in the DBpedia graph, using only the object properties. Heitmann et al. [116] propose an open recommender system which utilizes Linked Data to mitigate the new-user, new-item and sparsity problems of collaborative recommender systems. They first publish an existing music artists database as LOD using the FOAF ontology[1]. Then, they link the data to DBpedia and DBtune MySpace. Using the new connections between the users, artists and items, the authors are able to build a collaborative recommender system.

Di Noia et al. [64] propose a model-based recommender system that relies on LOD, and can use any arbitrary classifier to perform the recommendations. First,

[1] `http://xmlns.com/foaf/spec/`

they map the items from the local dataset to DBpedia, and then extract the direct property-value pairs. The resulting data is converted to feature vectors, where each property-value pair represents a feature. The work is extended in [65] where the similarities between the items are calculated using a vector space model. In this approach, for each item the direct property-value pairs are extracted from DBpedia, Freebase and LinkedMDB[2], which are represented as a 3-dimensional matrix where each slice refers to an ontology property and represents its adjacency matrix. In [206], the authors present SPrank, a hybrid recommendation algorithm for computing top-N item recommendations from implicit feedback exploiting the information available as LOD. In the approach, the authors try to extract features able to characterize the interactions between users, items and entities capturing the complex relationships between them. To do so, they extract all the paths that connect the user to an item in order to have a relevance score for that item, which are then used as features in the recommender algorithm. In more recent work [207], the authors propose a content-based recommender based on a neighborhood-based graph kernel, which computes semantic item similarities by matching their local neighborhood graphs.

Another approach by Schmachtenberg et al. [268] uses background knowledge from LinkedGeoData, to enhance a location-based recommendation system. The features for the recommendation system were generated using the FeGeLOD tool [225], the predecessor of the RapidMiner LOD extension.

Furthermore, several cross-domain recommender systems based on LOD data have been proposed in the literature. Fernández-Tobías et al. [86] proposed an approach that uses DBpedia as a cross-domain knowledge source for building a semantic network that links concepts from several domains. On such a semantic network, which has the form of a directed acyclic graph, a weight spreading activation algorithm [46] retrieves concepts in a target domain (music) that are highly related to other input concepts in a source domain (points of interest). The work is extended in [137, 138] by finding richer semantic relations between architecture and music concepts in DBpedia.

A similar LOD-enhanced graph-based approach is presented in [115, 117]. The approach is based on an enhanced spreading activation model that exploits intrinsic links between entities across a number of data sources.

## 11.2 Graph-based Methods for Recommender Systems

We consider three graph-based recommendation approaches. To perform the calculations, we first build an undirected weighted graph, where each item is represented as a node. For our implementation, we use the JUNG java library[3], which also offers implementations of different algorithms from graph theory. Since the domains for the three tasks differ, we use the same set of graph algorithms, but a different

[2] http://www.linkedmdb.org/

[3] http://jung.sourceforge.net/

graph of items with different edge weights for each of the tasks. The different graph algorithms are described in this section, while the dataset-specific construction of the respective graphs is described in the corresponding parts of Section 13.4.

### Shortest Path

To recommend relevant items for each user, we try to find the items in the graph that are closest to those items that were liked by the user, i.e., we assume that *proximity* in the graph is a proxy for *similarity*.

For implementing that approach, let $R$ be the set of items a user liked. Then, for each item $t$ in the test set (i.e., the items from which a recommendation is to be made), we compute the negated sum of shortest path lengths (given the edge weights) for all items in $R$ to $t$ as a ranking score:

$$I(t|R) = -\sum_{i=1}^{|R|} sp(R_i, t), \tag{11.1}$$

where $sp(R_i, t)$ is the shortest path from $R_i$ to $t$, where $R_i \in R$. Then, for each user the test items are sorted based on the relevance $I$ in descending order, and the top-N items are recommended to the user.

### K-Step Markov Approach

The PageRank algorithm is frequently used to compute importance for nodes in a graph. The PageRank score of a node can be seen as the probability of visiting that node with a random walk on the graph [208]. The K-Step Markov approach represents an algorithm variant of the PageRank algorithm with priors and computes the importance of any node in a given graph based on a given root set of nodes [327]. More precisely, the approach computes the relative probability that the system will spend time at any particular node in the graph, given that it starts in a root set of nodes *R* and ends after *K* steps.

The result is an estimate of the transient distribution of states in the Markov chain, starting from *R*: as *K* gets larger it will converge to the steady-state distribution used by PageRank, i.e. the standard version of PageRank without priors. Thus, the value of K controls the relative tradeoff between a distribution "biased" towards the root set of nodes *R* and the steady-state distribution which is independent of where the Markov process started. The relative importance of a node *t* given a root set *R* can be calculated using the equation:

$$I(t|R) = \left[A_{P_R} + A_{P_R}^2 + \ldots + A_{P_R}^K\right] \tag{11.2}$$

where *A* is the transition probability matrix of size $n \times n$, and $P_R$ is an $n \times 1$ vector of initial probabilities for the root set *R*.

In order to create recommendations, we again start with the set of items $R$ which a user likes, and then use the K-Step Markov approach to find the top-N

nodes that have the highest stationary probability. For the transition probability we use the edge weights after turning them into proper probabilities.

#### WeightedNIPaths

For predicting the relevance of an item node for a given user within the graph, we also make use of the WeightedNIPath algorithm [327]. Building upon the Shortest Path approach from Section 11.2, the idea is here to consider not only the single shortest path, but to take into account all distinct paths and add a decay factor $\lambda$ to penalize longer paths. Therefore, we compute the number of distinct paths between the source nodes, i.e. all nodes/items that have been rated $r \in R$ by the user, and each other node $t$, i.e. all unrated and potentially to be recommended nodes. Our intuition is that items which are frequently (indirectly) connected to positively rated items, have some content-based connection and should thus get a higher recommendation score by this method.

Formally, for a set of items $R$ liked by a user, and a candidate item $t$, we compute the importance score as

$$I(t|R) = \sum_{r \in R} \sum_{i=1}^{|P(r,t)|} \lambda^{-|p_i|} \tag{11.3}$$

where $P(r,t)$ is a set of maximum-sized node-disjoint paths from node $r$ to node $t$, $p_i$ is the $i$th path in $P(r,t)$, and $\lambda$ is the path decay coefficient. As before, the top-N items are recommended.

With this approach, movies that e.g. share the same actors and director will be closer related than two movies that have only the director in common. While being similar to the shortest path approach described above, WeightedNIPath takes into account all paths and discounts each edge of a path by a factor, thus penalizing longer paths (we use a discount factor of 3). The rationale here is that the longer a path, the less closely related are the items this path connects. In addition, based on initial experiments, we limit the path length to 2, which is also common practice when working with DBpedia as a semantic network [270].

### 11.2.1 Evaluation

We evaluate the item recommendation performance of the three graph algorithms with benchmark data from the Linked Open Data-enabled Recommender Systems Challenge 2015.[4] For this challenge, three training datasets from different domains, i.e., movies, books, and music, are provided. Those datasets were generated by collecting data from Facebook profiles about personal preferences ("likes") for the items. After a process of user anonymization, the items available in the dataset have been mapped to their corresponding DBpedia URIs. An overview of the size of the datasets is given in Table 11.1.

[4] `http://sisinflab.poliba.it/events/lod-recsys-challenge-2015/`

Table 11.1: Datasets Overview

| Dataset | #Items | #Ratings | Task |
|---|---|---|---|
| movies | 5,389 | 638,268 | 1 & 3 |
| music | 6,372 | 854,016 | 2 |
| books | 3,225 | 11,600 | 3 |

For all three tasks, each approach was evaluated on an unseen gold standard using an online evaluation system provided by the challenge, and compared to standard collaborative filtering as a baseline.

### Task 1: Top-N Recommendations from Unary User Feedback

In this task, top-N recommendations for the movie domain are to be made. The input is unary feedback (i.e., whether a user likes an item) under open world semantics, i.e., no negative examples (dislikes) are provided. The evaluation is made based on recall, precision, and F-measure for the top 10 recommendations.

### Graph Extraction

The graph we construct consists of some direct relations of the movies, as well as their actors, genres, and characters. To generate the graph, we used the RapidMiner Linked Open Data extension [229, 246]. We extracted the following relations:

- **movie:** *rdf:type*, *dcterms:subject*, *dbpedia-owl:starring*, *dbpedia-owl:director*, *dbpedia-owl:distributor*, *dbpedia-owl:producer*, *dbpedia-owl:musicComposer*, *dbpedia-owl:writer*, and *dbpprop:genre*
- **movie_actor:** *rdf:type*, *dcterms:subject*, and *is dbpedia-owl:starring of*
- **movie_genre:** *rdf:type* and *dcterms:subject*
- **movie_character:** *rdf:type*, *dcterms:subject*, *dbpedia-owl:creator*, and *dbpedia-owl:series*

Each DBpedia entity is represented as a node in the graph, where the relations between the entities are represented as undirected edges between the nodes in the graph. We use inverse document frequency (IDF) to weight the edges. For example, if an actor plays in five movies, then the IDF for each of those five relations is $log\frac{1}{5}$.[5]

In order to make the approach work also for rather weakly interlinked resources, we introduce additional edges in the graph based on the abstracts in DBpedia. To that end, we preprocess the abstract, i.e., we convert the abstract to lower

[5]To compute edge weights from IDF, we first normalize the IDF scores to $[0; 1]$, and then assign $1 - IDF_{normalized}$ as a weight to the edges, so that edges with a larger IDF value have a lower weight

case, perform tokenization, stemming, and stop words removal. Then, each token is represented as a node in the graph. Eventually, we use the resulting graph of abstract tokens and genre relations to find paths between entities from the movie domain and entities from the books domain. The relations between the entities and tokens are, as before, represented as undirected edges between the nodes in the graph and edges are also again IDF-weighted as for task 1.

In addition to the graph, we extracted the following *global* (i.e., not user-related) popularity scores:

- Number of Facebook likes[6]
- Metacritic score[7]
- Rotten Tomatoes score[8]
- DBpedia Global PageRank [295]
- Local Graph PageRank: Computed using the *JUNG* java library on the previously generated graph
- Aggregated Popularity: Using Borda's rank aggregation [54], we aggregated all those popularity scores into one.

we use a stacking approach [304] for combining all the recommendations, i.e., collaborative, content-based, and global.

### Results

The results are depicted in Table 11.2. It can be observed that the collaborative filtering based approaches clearly outperform the content-based ones. From the graph-based approaches, the shortest paths are the best performing approach.

The global ranking scores, aggregated with Borda's rank aggregation, are a strong competitor to content-based approaches. On the other hand, the stacking approach combining multiple recommendation approaches does not work well. This is in particular due to the fact that only positive evidence is given, which makes it hard to train a regression algorithm.

Due to its bad performance, we have not considered the stacking solution for the subsequent tasks.

### Task 2: Diversity within Recommended Item Sets

The second task is to make recommendations in the music domain. Here, the focus is on *diverse* recommendations, i.e., entities from different genres. The evaluation is made based on the average of F-measure and intra-list diversity (ILD) for the top 20 recommendations.

[6] `https://www.facebook.com/`
[7] `http://www.metacritic.com/`
[8] `http://www.rottentomatoes.com/`

Table 11.2: Results for top-N recommendations from unary user feedback (the best results are marked in bold).

| Approach | P@10 | R@10 | F1@10 |
|---|---|---|---|
| User-Based KNN (k=20) | 0.0954 | 0.1382 | 0.1129 |
| User-Based KNN (k=50) | 0.1025 | 0.1485 | 0.1213 |
| User-Based KNN (k=80) | **0.1032** | **0.1493** | **0.122** |
| Shortest Path | 0.0597 | 0.0859 | 0.0704 |
| K-Step Markov Approach (K=4) | 0.0496 | 0.0703 | 0.0581 |
| WeightedNIPaths (H=2) | 0.0151 | 0.0217 | 0.0178 |
| Shortest Path (w abstract) | 0.0525 | 0.0746 | 0.0616 |
| K-Step Markov Approach (K=4) (w abstract) | 0.0562 | 0.0804 | 0.0662 |
| WeightedNIPaths (H=2) (w abstract) | 0.0216 | 0.0309 | 0.0254 |
| Borda's rank aggregation | 0.0572 | 0.0824 | 0.0676 |
| Stacking with polynomial regression | 0.0099 | 0.0137 | 0.0115 |

### Graph Extraction

To generate the graph, we extracted the following relations:

- **music_artist:** *rdf:type*, *dcterms:subject*, *dbpedia-owl:genre*, *dbpedia-owl:associatedBand*, *dbpedia-owl:associatedMusicalArtist*, *dbpedia-owl:genre*, and *dbpedia-owl:occupation*
- **music_band:** *rdf:type*, *dcterms:subject*, *dbpedia-owl:associatedBand*, *dbpedia-owl:associatedMusicalArtist*, *dbpedia-owl:genre*, and *dbpedia-owl:bandMember*
- **music_album:** *rdf:type*, *dcterms:subject*, *dbpedia-owl:artist*, and *dbpedia-owl:genre*
- **music_composition:** *rdf:type*, *dcterms:subject*, *dbpedia-owl:musicalArtist*, *dbpedia-owl:musicalBand*, and *dbpedia-owl:genre*
- **music_genre:** *rdf:type* and *dcterms:subject*
- **abstract:** *dbpedia-owl:abstract*

As for the previous task, each DBpedia entity is represented as a node in the graph, where the relations between the entities are represented as undirected edges between the nodes in the graph. Like for the previous task, we use IDF to weight the edges.

For making the predictions, with user-based k-NN, we simply predict the top 20 items, as we already observe a high ILD with this approach. For shortest paths and the k-step Markov approach: We first generate ranked lists. From those lists, we then pick the top 10 items, and then iteratively fill them up with the next 10 items in the list that do not share a genre with those that are already in the list.

Table 11.3: Results for diversity within recommended item sets (the best results are marked in bold).

| Approach | P@20 | R@20 | F1@20 | ILD@20 |
|---|---|---|---|---|
| User-Based KNN (k=20) | 0.09 | 0.2477 | 0.1321 | 0.9039 |
| User-Based KNN (k=50) | 0.0963 | 0.2649 | 0.1412 | 0.9028 |
| User-Based KNN (k=80) | **0.0973** | **0.2677** | **0.1427** | 0.9032 |
| Shortest Path | 0.0343 | 0.0948 | 0.0504 | 0.8964 |
| K-Step Markov Approach (K=4) | 0.0312 | 0.0863 | 0.0458 | 0.9077 |
| WeightedNIPaths (H=2) | 0.0343 | 0.0933 | 0.0502 | 0.8588 |
| Shortest Path (w abstract) | 0.0077 | 0.0203 | 0.0112 | **0.9717** |
| K-Step Markov Approach (K=4) (w abstract) | 0.0217 | 0.0585 | 0.0317 | 0.9699 |
| WeightedNIPaths (H=2) (w abstract) | 0.0358 | 0.0975 | 0.0523 | 0.8785 |
| Borda's rank aggregation | 0.0356 | 0.0977 | 0.0522 | 0.922 |

#### Results

The results for task 2 are depicted in Table 11.3. Again, we can see that on average, the collaborative filtering approaches produce the better results in terms of F-measure, with the ILD being comparable. It is furthermore remarkable that the ILD is that high for the collaborative filtering approaches, which were not specifically altered for producing diverse recommendations. The highest ILD score is achieved with the Shortest Path (with abstract), however at the cost of a very low recall, precision, and F1 score.

### Task 3: Cross-domain Recommendation

The third task poses a different setting, as it asks for using feedback (user ratings) from one domain, here movies, to provide recommendations for another domain, namely books. Like for the first task, recall, precision, and F-measure for the top 10 recommendations are used as evaluation metrics.

#### Graph Extraction

To generate the graph, we extract the same features as for the top-N recommendation tasks on movies (see Section 11.2.1), but including in addition the *dbpedia-owl:abstract* for each item. For the book domain, we extract the following relations:

- **book:** *rdf:type*, *dcterms:subject*, *dbpedia-owl:genre*, *dbpedia-owl:author*, and *dbpedia-owl:subsequentWork*
- **book_writer:** *rdf:type* and *dcterms:subject*
- **book_character:** *rdf:type* and *dcterms:subject*

Table 11.4: Results for cross-domain recommendation (the best results are marked in bold).

| Approach | P@10 | R@10 | F1@10 |
|---|---|---|---|
| User-Based KNN (k=20) | 0.0162 | 0.026 | 0.0199 |
| User-Based KNN (k=50) | 0.022 | 0.0353 | 0.0271 |
| User-Based KNN (k=80) | 0.0258 | 0.0416 | 0.0318 |
| Shortest Path | 0.0326 | 0.0539 | 0.0407 |
| K-Step Markov Approach (K=4) | 0.0659 | 0.1077 | 0.0818 |
| WeightedNIPaths (H=2) | 0.0358 | 0.0299 | 0.0227 |
| Shortest Path (w abstract) | 0.0627 | 0.1026 | 0.0778 |
| K-Step Markov Approach (K=4) (w abstract) | **0.078** | **0.1276** | **0.0968** |
| WeightedNIPaths (H=2) (w abstract) | 0.0195 | 0.0314 | 0.024 |
| Borda's rank aggregation | 0.0301 | 0.0493 | 0.0374 |

- **book_genre:** *rdf:type* and *dcterms:subject*
- **abstract:** *dbpedia-owl:abstract*

#### Results

The results for task 3 are depicted in Table 11.4. Here, in contrast to task 1 and 2, we find that the graph-based approaches clearly outperform the collaborative filtering (CF) ones. We also observe that User-based KNN is comparably low, which leads us to the suspicion that the cross-domain nature of this task poses a serious challenge to regular CF approaches – which obviously need to operate on already observed items. In an extreme cross-domain scenario, there would be no historical user preference information available on the items to be ranked and any CF approach would fail. In contrast, the graph-based approaches can apparently find reasonable relations in the graph between books and movies (e.g., common genres, or mentioning of similar terms in the abstract) and leverage those for creating meaningful predictions.

### 11.2.2 Conclusion and Outlook

In this section, we have proposed to derive weighted graphs from DBpedia, and apply graph algorithms on it to retrieve item-based recommendations. We studied the usage of three different graph algorithms working on different subgraphs of the DBpedia graph. Our approaches rely on (a) shortest paths, (b) a variant of PageRank with priors (K-Step Markov), and (c) the sum of all distinct, connecting paths (WeightedNIPaths).

We find that in situations where the complete user feedback is available, collaborative filtering outperforms all studied graph-based approaches. In contrast,

in situations where user feedback is scarce – here: for making cross-domain predictions – graph-based approaches are a reasonable way to build recommender systems. This observation makes the approaches proposed in this section an interesting candidate for various settings. For example, in cold start situations, where no ratings for new products exist (yet), they should be included in recommendations. Second, when trying to open new market segments for an existing customer base, such methods can be helpful.

So far, we have considered only DBpedia as LOD source. In future work we can explore the existing *owl:sameAs* links in DBpedia to build richer and denser graphs from domain specific LOD sources, e.g., LinkedMDB[9] for movies, MusicBrainz[10] for music, the British National Bibliography[11] for books, etc. The information retrieved from the domain specific LOD sources should be more accurate and extensive, which should lead to better performance of the recommender systems. Furthermore, when building the graphs only specific relations were included, which we believed were the most relevant for the task. However, the graphs may be built in unsupervised manner, i.e., including all properties for all of the entities, and expanding the graph to several hops. The proposed approaches should still be able to make good recommendations on such graphs, because we use IDF to weight the edges, i.e., the most relevant edges will have a higher weight. This way, we would be able to apply the approaches on data from any domain/s without the need for manual feature engineering.

## 11.3 A Hybrid Multi-Strategy Recommender System Using Semantic Web Knowledge Graphs

As a second recommender system, we propose a hybrid, multi-strategy approach that combines the results of different base recommenders and generic recommenders into a final recommendation. A *base recommender* is an individual collaborative or content based recommender system, whereas a *generic recommender* makes a recommendation solely on some global popularity score, which is the same for all users. The approach has been evaluated on the three tasks of the *LOD-enabled Recommender Systems Challenge 2014* from the domain of book recommendations.[12] For base recommenders, we use two collaborative filtering strategies (item and user based), as well as different content-based strategies exploiting various feature sets created from DBpedia.

[9] `http://www.linkedmdb.org/`

[10] `https://wiki.musicbrainz.org/LinkedBrainz`

[11] `http://bnb.data.bl.uk/`

[12] 75,559 numeric ratings on 6,166 books (from 0-5, Task 1) and 72,372 binary ratings on 6733 books (Tasks 2 and 3), resp., from 6,181 users for training, and evaluation on 65,560 and 67,990 unknown ratings, resp. See `http://challenges.2014.eswc-conferences.org/index.php/RecSys` for details.

### Generic Recommenders

We use different generic recommenders in our approach. First, the RDF Book Mashup dataset[13] provides the average score assigned to a book on Amazon. Furthermore, DBpedia provides the number of ingoing links to the Wikipedia article corresponding to a DBpedia instance, and the number of links to other datasets (e.g., other language editions of DBpedia), which we also use as global popularity measures. Finally, SubjectiveEye3D delivers a subjective importance score computed from Wikipedia usage information.[14]

### Features for Content-based Recommendation

The features for content-based recommendation were extracted from DBpedia using the RapidMiner Linked Open Data extension [229]. We use the following feature sets for describing a book:

- All *direct types*, i.e., `rdf:type`, of a book[15]
- All *categories of a book*
- All *categories of a book including broader categories*[16]
- All *categories of a book's author(s)*
- All *categories of a book's author(s) and of all other books* by the book's authors
- All *genres of a book* and of all other books by the book's authors
- All *authors that influenced or were influenced* by the book's authors
- A bag of words created from the *abstract* of the book in DBpedia. That bag of words is preprocessed by tokenization, stemming, removing tokens with less than three characters, and removing all tokens less frequent than 3% or more frequent than 80%.

Furthermore, we created a *combined book's feature set*, comprising direct types, qualified relations, genres and categories of the book itself, its previous and subsequent work and the author's notable work, the language and publisher, and the bag of words from the abstract. Table 11.5 depicts the number of features in each set.

Besides DBpedia, we made an effort to retrieve additional features from two additional LOD sources: British Library Bibliography and DBTropes[17]. Using the

[13] `http://wifo5-03.informatik.uni-mannheim.de/bizer/bookmashup/`

[14] `https://github.com/paulhoule/telepath/wiki/SubjectiveEye3D`

[15] This includes types in the YAGO ontology, which can be quite specific (e.g., *American Thriller Novels*)

[16] The reason for not including broader categories by default is that the category graph is not a cycle-free tree, with some subsumptions being rather questionable.

[17] `http://bnb.data.bl.uk/` and `http://skipforward.opendfki.de/wiki/DBTropes`

RapidMiner LOD extension, we were able to link more than 90% of the books to BLB entities, but only 15% to DBTropes entities. However, the generated features from BLB were redundant with the features retrieved from DBpedia, and the coverage of DBTropes was too low to derive meaningful features. Hence, we did not pursue those sources further.

### Recommender Strategies

For implementing the collaborative and content-based recommendation systems, we used the RapidMiner Recommendation Extension [176], which uses k-NN classification. We use $k = 80$ and cosine similarity for the base recommenders. The rationale of using cosine similarity is that, unlike, e.g., Euclidean distance, only common features influence the similarity, but not common absence of features (e.g., two books *not* being American Thriller Novels).

Furthermore, we train an additional recommender on the joint feature set, using Random Decision Trees (RDTs) [336].[18] RDTs generate $k_1$ decision trees with maximal depth $k_2$ and random attribute tests at the inner nodes. Each tree collects a distribution over the target variables at each of its leaf nodes by seeing the training data. E.g. for multilabel data, RDT's leaves collect the label distribution so that each RDT predicts for each test instance a distribution over the labels. These predictions are subsequently averaged over all trees in order to produce one single prediction. The predictions of several of such trees are then combined into a final prediction. RDTs provide a good tradeoff between scalability for large example sets and prediction accuracy (often outperforming SVMs).

For applying RDTs to the collaborative filtering data, we transformed the problem into a multilabel task: For each user we generated $n$ different labels indicating each of the possible user ratings, i.e. $n = 5$ for task 1 and $n = 2$ for task 2. During training RDTs learn – for each known book/user combination – the mapping between the feature set of each book and the generated labels. Given an unknown book/user combination $x, y$, we are now able to estimate a distribution $P(i|\ x, y)$ over the different ratings $i$. The final predicted rating $r$ is obtained by weighting the ratings $r = \sum_{i=0}^{5} i \cdot P(i|\ x, y)$ (task 1) or by computing the probability difference $P(1|\ x, y) - P(0|\ x, y)$ (task 2).

RDTs do not suffer from high dimensionality and sparseness as much as k-NN does, thus we have built $k_1 = 10$ trees with depth $k_2 = 10$ on the combined book's properties feature set, instead of individual RDTs on each feature set.[19]

[18] We used the implementation available at `http://www.dice4dm.com/`

[19] In general, it holds that the higher $k_1$ and $k_2$ the better, since this increases the number of covered feature dimensions and the diversity of the ensemble. However, comparably small values of $k_1$ and $k_2$, around 10 or 20 and maximally 100, are sufficient according to experiments by Zhang et al. [336] and Kong and Yu [152]. In our experiments, we tried to find a good balance between computational costs and predictive quality, and we report the combination which we used for our final recommendations.

Table 11.5: Performances of the base and generic recommenders, the number of features used for each base recommender, and the performance of the combined recommenders

| | | Task 1 | | Task 2 |
|---|---|---|---|---|
| **Recommender** | **#Features** | **RMSE** | **LR $\beta$** | **F-Score** |
| *Item-based collaborative filtering* | – | 0.8843 | +0.269 | 0.5621 |
| *User-based collaborative filtering* | – | 0.9475 | +0.145 | 0.5483 |
| *Book's direct types* | 534 | 0.8895 | -0.230 | 0.5583 |
| *Author's categories* | 2,270 | 0.9183 | +0.098 | 0.5576 |
| *Book's (and author's other books') genres* | 582 | 0.9198 | +0.082 | 0.5567 |
| *Combined book's properties* | 4,372 | 0.9421 | +0.0196 | 0.5557 |
| *Author and influenced/influencedBy authors* | 1,878 | 0.9294 | +0.122 | 0.5534 |
| *Books' categories and broader categories* | 1,987 | 0.939 | +0.012 | 0.5509 |
| *Abstract bag of words* | 227 | 0.8893 | +0.124 | 0.5609 |
| *RDT recommender on combined book's properties* | 4,372 | 0.9223 | +0.128 | 0.5119 |
| *Amazon rating* | – | 1.037 | +0.155 | 0.5442 |
| *Ingoing Wikipedia links* | – | 3.9629 | +0.001 | 0.5377 |
| *SubjectiveEye3D score* | – | 3.7088 | +0.001 | 0.5369 |
| *Links to other datasets* | – | 3.3211 | +0.001 | 0.5321 |
| ***Average of all individual recommenders*** | 14 | 0.8824 | – | – |
| ***Stacking with linear regression*** | 14 | 0.8636 | – | 0.4645 |
| ***Stacking with RDT*** | 14 | **0.8632** | – | 0.4966 |
| ***Borda rank aggregation*** | 14 | – | – | **0.5715** |

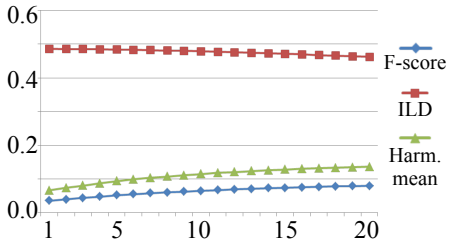
### 11.3.1 Predicting Ratings and Top k Lists

For predicting ratings (task 1 in the challenge), we use all the recommendation algorithms discussed above for training a regression model in the range of $[0; 5]$. The results for the base and generic recommenders are shown in Fig. 11.5.

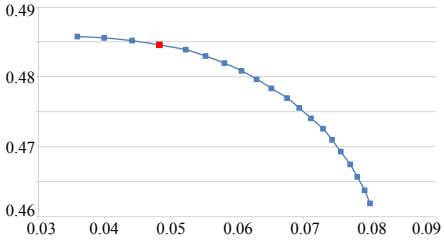
In order to create a more sophisticated combination of those recommenders, we trained a *stacking* model as described in [304]: We trained the base recommenders in 10 rounds in a cross validation like setting, collected their predictions, and learned a stacking model on the predictions. The results in Table 11.5 show that the stacked prediction outperforms the base and generic recommenders, with the RDT based stacking (with $k_1 = 500$ and $k_2 = 20$) slightly ahead of linear regression, and both stacking approaches outperforming the baseline approach of averaging all recommenders' ratings.

To further analyze the contribution of each feature, we also report the $\beta$ parameters found by linear regression. It can be observed that apart from the direct types, all base and generic recommenders contribute to the linear regression. A possible reason for that anomaly is that direct types and categories are rather redundant. Furthermore, we can see the benefit of using stacking approaches as the three generic recommenders with high RMSE are filtered out by the LR model.

For creating top k lists from binary ratings (task 2 in the challenge), we again trained regression models like for rating prediction, using a range of $[0; 1]$. The top k lists were then obtained by ranking by the predicted rating. As shown in Table 11.5, the base recommenders worked quite well, but the combination with




(a) F-measure, ILD and their harmonic mean ($y$-axis) for different values of $m$




(b) ILD ($y$-axis) plotted against F-measure ($x$-axis)

Figure 11.1: Trade-off between F-measure and diversity

linear regression delivered non-satisfying results. The reason is that the outcome of the base recommenders is not scaled equally for each user, but strongly depends on the user's total number of positive and negative ratings. This made it impossible to learn a suitable regression function.

However, we observed that despite being incompatible in scale, the base and generic recommenders delivered good *rankings* for each user. Thus, we performed an aggregation of the rankings produced by the different recommenders, using Borda's rank aggregation algorithm, which outperforms all the individual recommenders, as well as the stacking regression.

### 11.3.2 Creating Diverse Predictions

The final task in the challenge was to address *diversity* of predictions, i.e., trade off the accuracy of predictions, measured in F1 score, and their diversity, measured in intra-list diversity (ILD), both on a top k list. To address that trade-off, we followed a greedy top down approach which creates a ranking as for top k lists. First, we select the top $m$ items from that list. Then, we process the list from position $m+1$ on, adding each book that does not share author and categories with any of the books already on the list, until the list has $k$ items.

The results are depicted in Fig. 11.1 for k=20, selecting items from a list of the top 100 predictions. It can be observed that the F1 score gradually rises when using higher values of $m$, while the ILD drops. Although the harmonic mean is optimal for using simply the top 20 predictions (given the different orders of magnitude of F1 and ILD), we decided to submit the solution with $m = 4$ to the challenge.[20]

### 11.3.3 Conclusion and Outlook

In this section, we have laied out a hybrid multi-strategy approach for linked data enabled recommender systems. We have shown that combining the predictions of

[20]The reason is that the challenge uses the average rank w.r.t. F1 and ILD as a scoring function, which makes the selection of an optimal parameter strongly depend on the other participants' solutions. It turned out that $m = 4$ optimized our scoring.

different base recommenders is a feasible strategy, and that generic (i.e., non user specific) recommenders can be a useful ingredient.

In particular, our approach allows for the addition of new feature groups without interaction effects, and for the combination of different recommender strategies. By exploiting stacking regression, an optimal combination of different recommenders can be found automatically, however, for ranking-based problems, rank aggregation turned out to be the more promising strategy.

## 11.4 A Content-Based Recommender System Using Semantic Web Knowledge Graph Embeddings

One of the main limitations of traditional content-based recommendation approaches is that the information on which they rely is generally insufficient to elicit user's interests and characterize all the aspects of their interaction with the system. This is the main drawback of the approaches built on textual and keyword-based representations, which cannot capture complex relations among objects since they lack the semantics associated to their attributes. A process of "knowledge infusion" [275] and semantic analysis has been proposed to face this issue, and numerous approaches that incorporate ontological knowledge have been proposed, giving rise to the newly defined class of *semantics-aware* content-based recommender systems [56].

### 11.4.1 Approach

In this approach we use the RDF2vec graph embeddings technique for feature generation in the context of content-based RS, and relies on a relatively simple recommendation algorithm, i.e., the item-based K-Nearest Neighbor approach [245] with cosine similarity. Formally, this method evaluates the closeness of items through cosine similarity between the corresponding features vectors and then selects a subset of those – the neighbors – for each item, that will be used to estimate the rating of user $u$ for a new item $i$ as follows:

$$r^*(u,i) = \frac{\sum_{j \in ratedItems(u)} cosineSim(j,i) \cdot r_{u,j}}{\sum_{j \in ratedItems(u)} |cosineSim(j,i)|} \tag{11.4}$$

where $ratedItems(u)$ is the set of items already evaluated by user $u$, $r_{u,j}$ indicates the rating for item $j$ by user $u$ and $cosineSim(j,i)$ is the cosine similarity score between items $j$ and $i$. In our experiments, the size of the considered neighbourhood is limited to 5.

### 11.4.2 Experiments

We evaluate different variants of our approach on three datasets, and compare them to common approaches for creating content-based item representations from LOD,

Table 11.6: Statistics about the three datasets

| | **Movielens** | **LibraryThing** | **Last.fm** |
|---|---|---|---|
| Number of users | 4,186 | 7,149 | 1,875 |
| Number of items | 3,196 | 4,541 | 2,432 |
| Number of ratings | 822,597 | 352,123 | 44,981 |
| Data sparsity | 93.85% | 98.90% | 99.01% |

as well as to state of the art collaborative and hybrid approaches. Furthermore, we investigate the use of two different knowledge graphs, i.e., DBpedia and Wikidata.

**Datasets** In order to test the effectiveness of vector space embeddings for the recommendation task, we have performed an extensive evaluation in terms of ranking accuracy on three datasets belonging to different domains, i.e., `Movielens`[21] for movies, `LibraryThing`[22] for books, and `Last.fm`[23] for music. The first dataset, `Movielens 1M`, contains 1 million 1-5 stars ratings from 6,040 users on 3,952 movies. The dataset `LibraryThing` contains more than 2 millions ratings from 7,279 users on 37,232 books. As in the dataset there are many duplicated ratings, when a user has rated the same item more than once, her last rating is selected. The unique ratings are 749,401, in the range from 1 to 10. Both `Movielens` and `LibraryThing` datasets contain explicit ratings, and to test the approach also on implicit feedbacks, a third dataset built on the top of the `Last.fm` music system is considered. `Last.fm` contains 92,834 interactions between 1,892 users and 17,632 musical artists. Each interaction is annotated with the corresponding listening count.

The original datasets are enriched with background information using the item mapping and linking to DBpedia technique described in [205], whose dump is available at `https://github.com/sisinflab/LODrecsys-datasets`. Since not all the items have a corresponding resource in DBpedia, after the mapping, the versions of `Movielens`, `LibraryThing` and `Last.fm` datasets contain 3,883 movies, 11,695 books, and 11,180 musical artists, respectively.

The datasets are finally preprocessed to guarantee a fair comparison with the state of the art approaches described in [62]. Here, the authors propose to (i) remove popularity biases from the evaluation not considering the top 1% most popular items, (ii) reduce the sparsity of `Movielens` dataset in order to have at least a sparser test dataset and (iii) remove from `LibraryThing` and `Last.fm` users with less than five ratings and items rated less than five times. The final statistics on the three datasets are reported in Table 11.6.

[21] `http://grouplens.org/datasets/movielens/`
[22] `https://www.librarything.com/`
[23] `http://www.lastfm.com`

Table 11.7: Results of the ItemKNN approach on `Movielens` dataset with reference to different computation of features.

| Strategy | Precision | Recall | F1 | nDCG |
|---|---|---|---|---|
| types | 0.00313 | 0.00145 | 0.00198 | 0.28864 |
| categories | 0.0305 | 0.02093 | 0.02482 | 0.30444 |
| rel in | 0.01122 | 0.00589 | 0.0077 | 0.29183 |
| rel out | 0.02844 | 0.01607 | 0.02053 | 0.30274 |
| rel in & out | 0.02852 | 0.01566 | 0.02021 | 0.3006 |
| rel-vals in | 0.03883 | 0.02293 | 0.02882 | 0.29411 |
| rel-vals out | 0.01279 | 0.00971 | 0.011 | 0.29378 |
| rel-vals in & out | 0.01174 | 0.00913 | 0.01027 | 0.29333 |
| WC_4 | 0.00684 | 0.00343 | 0.0045 | 0.29032 |
| WL_2_2 | 0.00601 | 0.00288 | 0.00389 | 0.28977 |
| DB_TransE | 0.03047 | 0.01411 | 0.01928 | 0.30385 |
| DB_TransH | 0.02649 | 0.01187 | 0.01639 | 0.30016 |
| DB_TransR | 0.00941 | 0.0043 | 0.0059 | 0.29216 |
| DB2vec SG 200w 200v 4d | 0.05423 | 0.02693 | 0.03598 | 0.31676 |
| DB2vec CBOW 200w 200v 4d | 0.03475 | 0.01637 | 0.02225 | 0.30426 |
| DB2vec CBOW 500w 200v 4d | 0.03893 | 0.02167 | 0.02784 | 0.30782 |
| DB2vec CBOW 500w 500v 4d | 0.03663 | 0.02088 | 0.02659 | 0.30557 |
| DB2vec SG 500w 200v 4d | 0.05681 | **0.03119** | **0.04027** | **0.31828** |
| DB2vec SG 500w 500v 4d | **0.05786** | 0.0304 | 0.03985 | 0.31726 |
| DB2vec CBOW 500w 200v 8d | 0.01064 | 0.00548 | 0.00723 | 0.29245 |
| DB2vec CBOW 500w 500v 8d | 0.01137 | 0.00567 | 0.00756 | 0.29289 |
| DB2vec SG 500w 200v 8d | 0.04424 | 0.02693 | 0.03347 | 0.30997 |
| DB2vec SG 500w 500v 8d | 0.02191 | 0.01478 | 0.01765 | 0.29863 |
| WD2vec CBOW 200w 200v 4d | 0.01217 | 0.00596 | 0.00800 | 0.29362 |
| WD2vec CBOW 200w 500v 4d | 0.01027 | 0.00427 | 0.0060 | 0.29211 |
| WD2vec SG 200w 200v 4d | 0.02902 | 0.01479 | 0.01959 | 0.30189 |
| WD2vec SG 200w 500v 4d | 0.02644 | 0.01246 | 0.01693 | 0.29967 |

### Evaluation Protocol

The ranking setting for the recommendation task consists of producing a ranked list of items to suggest to the user and in practical situations turns into the so-called top-$N$ recommendation task, where just a cut-off of the ranked list of size $N$ is provided to the user. This setting has recently replaced the rating prediction, because of the increasing awareness that the user is not interested in an accurate prediction of the item rating, but is looking for a (limited) list of items extracted from the pool of available ones.

As evaluation ranking protocol for our comparison, we adopted the *all unrated items* methodology presented in [288] and already used in [62]. Such methodology asks to predict a score for each item not rated by a user, irrespective of the existence of an actual rating, and to compare the recommendation list with the test set.

Table 11.8: Results of the ItemKNN approach on `LibraryThing` dataset with reference to different computation of features.

| Strategy | Precision | Recall | F1 | nDCG |
|---|---|---|---|---|
| types | 0.01854 | 0.04535 | 0.02631 | 0.16064 |
| categories | 0.06662 | 0.15258 | 0.09274 | 0.23733 |
| rel in | 0.04577 | 0.10219 | 0.06322 | 0.20196 |
| rel out | 0.04118 | 0.09055 | 0.05661 | 0.19449 |
| rel in & out | 0.04531 | 0.10165 | 0.06268 | 0.20115 |
| rel-vals in | 0.06176 | 0.14101 | 0.08589 | 0.22574 |
| rel-vals out | 0.06163 | 0.13763 | 0.08513 | 0.22826 |
| rel-vals in & out | 0.06087 | 0.13662 | 0.08421 | 0.22615 |
| WC_4 | 0.00159 | 0.00306 | 0.00209 | 0.12858 |
| WL_2_2 | 0.00155 | 0.00389 | 0.00221 | 0.12937 |
| DB_TransE | 0.01819 | 0.04705 | 0.02623 | 0.1585 |
| DB_TransH | 0.01466 | 0.03997 | 0.02145 | 0.15331 |
| DB_TransR | 0.00162 | 0.00341 | 0.00219 | 0.12947 |
| DB2vec SG 200w 200v 4d | 0.00442 | 0.00942 | 0.00601 | 0.13502 |
| DB2vec CBOW 200w 200v 4d | 0.00466 | 0.00933 | 0.00621 | 0.13595 |
| DB2vec CBOW 500w 200v 4d | 0.05127 | 0.11777 | 0.07143 | 0.21244 |
| DB2vec CBOW 500w 500v 4d | 0.05065 | 0.11557 | 0.07043 | 0.21039 |
| DB2vec SG 500w 200v 4d | 0.05719 | 0.12763 | 0.07898 | 0.2205 |
| DB2vec SG 500w 500v 4d | 0.05811 | 0.12864 | 0.08005 | 0.22116 |
| DB2vec CBOW 500w 200v 8d | 0.00836 | 0.02334 | 0.01231 | 0.14147 |
| DB2vec CBOW 500w 500v 8d | 0.00813 | 0.02335 | 0.01206 | 0.14257 |
| DB2vec SG 500w 200v 8d | **0.07681** | **0.17769** | **0.10725** | **0.25234** |
| DB2vec SG 500w 500v 8d | 0.07446 | 0.1743 | 0.10434 | 0.24809 |
| WD2vec CBOW 200w 200v 4d | 0.00537 | 0.01084 | 0.00718 | 0.13524 |
| WD2vec CBOW 200w 500v 4d | 0.00444 | 0.00984 | 0.00611 | 0.13428 |
| WD2vec SG 200w 200v 4d | 0.06416 | 0.14565 | 0.08907 | 0.23309 |
| WD2vec SG 200w 500v 4d | 0.06031 | 0.14194 | 0.08465 | 0.22752 |

The metrics involved in the experimental comparison are three well-known ranking measures for recommendation accuracy, i.e., precision, recall, F-score (F1) and nDCG.

- *precision@N* [245] represents the fraction of relevant items in the top-$N$ recommendations.

- *recall@N* [245] indicates the fraction of relevant items, in the user test set, occurring in the top-$N$ list. As relevance threshold, we set 4 for `Movielens` and 8 for `LibraryThing`, as previously done in [62].

- The *F1* score [245], i.e., the harmonic mean of precision and recall, is also pointed out to be thorough. Although precision and recall are good indicators to evaluate the accuracy of a recommendation engine, they are not rank-sensitive.

Table 11.9: Results of the ItemKNN approach on `Last.fm` with reference to different computation of features.

| Strategy | Precision | Recall | F1 | nDCG |
|---|---|---|---|---|
| types | 0.00525 | 0.03256 | 0.009 | 0.01826 |
| categories | 0.01762 | 0.09889 | 0.02991 | 0.06023 |
| rel in | 0.00625 | 0.03625 | 0.01066 | 0.02042 |
| rel out | 0.00519 | 0.02757 | 0.00873 | 0.01733 |
| rel in & out | 0.00718 | 0.04205 | 0.01226 | 0.02567 |
| rel-vals in | 0.0185 | 0.10502 | 0.03145 | 0.06733 |
| rel-vals out | 0.00805 | 0.04585 | 0.01369 | 0.0248 |
| rel-vals in & out | 0.00339 | 0.01774 | 0.00569 | 0.00982 |
| WC_4 | 0.00086 | 0.00401 | 0.00141 | 0.00241 |
| WL_2_2 | 0.00086 | 0.00344 | 0.00137 | 0.00215 |
| DB_TransE | 0.01117 | 0.06078 | 0.01887 | 0.03953 |
| DB_TransH | 0.01409 | 0.07928 | 0.02392 | 0.04881 |
| DB_TransR | 0.0011 | 0.00523 | 0.00181 | 0.00381 |
| DB2vec SG 200w 200v 4d | 0.01 | 0.05498 | 0.01692 | 0.02911 |
| DB2vec CBOW 200w 200v 4d | 0.00929 | 0.05227 | 0.01577 | 0.03288 |
| DB2vec CBOW 500w 200v 4d | 0.01749 | 0.09915 | 0.02973 | 0.06435 |
| DB2vec CBOW 500w 500v 4d | 0.01769 | 0.10016 | 0.03006 | 0.06404 |
| DB2vec SG 500w 200v 4d | 0.02015 | 0.11109 | 0.03411 | 0.07232 |
| DB2vec SG 500w 500v 4d | 0.02001 | 0.10978 | 0.03385 | 0.07448 |
| DB2vec CBOW 500w 200v 8d | 0.00944 | 0.05349 | 0.01604 | 0.03311 |
| DB2vec CBOW 500w 500v 8d | 0.00964 | 0.0563 | 0.01646 | 0.03166 |
| DB2vec SG 500w 200v 8d | **0.0234** | **0.1359** | **0.03992** | **0.08719** |
| DB2vec SG 500w 500v 8d | 0.02088 | 0.12248 | 0.03567 | 0.07789 |
| WD2vec CBOW 200w 200v 4d | 0.00133 | 0.00785 | 0.00227 | 0.00382 |
| WD2vec CBOW 200w 500v 4d | 0.001 | 0.00532 | 0.00168 | 0.00408 |
| WD2vec SG 200w 200v 4d | 0.00612 | 0.03388 | 0.01036 | 0.02157 |
| WD2vec SG 200w 500v 4d | 0.00658 | 0.03932 | 0.01127 | 0.02382 |

- The normalized Discounted Cumulative Gain *nDCG@*$N$ [11] instead takes into account also the position in the recommendation list, being defined as

$$\mathrm{nDCG@}\,N = \frac{1}{\mathrm{iDCG}} \cdot \sum_{i=1}^{N} \frac{2^{rel(u,i)} - 1}{log_2(1+i)} \tag{11.5}$$

  where $rel(u, i)$ is a boolean function representing the relevance of item $i$ for user $u$ and iDCG is a normalization factor that sets $\mathrm{nDCG@}\,N$ value to 1 when an ideal ranking is returned [11].

As suggested in [288] and set up in [62], in the computation of nDCG@$N$ we fixed a default "neutral" value for those items with no ratings, i.e., 3 for `Movielens` and 5 for `LibraryThing`.

Table 11.10: Examples of K-Nearest Neighbor sets on Movielens.

| Query Movie | K Nearest Neighbours |
|---|---|
| Batman | Batman Forever, Batman Returns, Batman & Robin, Superman IV: The Quest for Peace, Dick Tracy |
| Bambi | Cinderella, Dumbo, 101 Dalmatians , Pinocchio, Lady and the Tramp |
| Star Trek: Generations | Star Trek VI: The Undiscovered Country, Star Trek: Insurrection, Star Trek III: The Search for Spock, Star Trek V: The Final Frontier, Star Trek: First Contact |

All the results have been computed @10, that is considering the top-10 list recommended to each user and then averaging across all users.

To evaluate the approach we use the RDF2vec graph embeddings (described in Chapter 8), built on the DBpedia and Wikidata knowledge graph. We compare our approach to the same baselines used in the evaluation section of Chapter 8.

### Results

We present the results by a purely content-based RS using RDF2Vec.

Tables 11.7, 11.8 and 11.9 contain the values of precision, recall, F-score (F1) and nDCG, respectively for Movielens, LibraryThing and Last.fm. The computation of recommendations has been done with the publicly available library RankSys.[24]

The first conclusion that can be drawn from Tables 11.7, 11.8 and 11.9 is that the best approach for all datasets is retrieved with a skip-gram model, 500 walks per entity and with a size of 200 for vectors built upon DBpedia. Altough on Movielens, the highest value of precision is achived using vector size of 500, the size 200 is prevalent according to the F1 measure. A substantial difference concerns the exploratory depth of the random walks, since for Movielens the results related to a depth of 4 outperform those computed with a depth of 8, while the tendency is reversed for both LibraryThing and Last.fm. Secondly, the advantage of the Skip-Gram model over CBOW is a constant both on DBpedia and Wikidata and is particularly evident when the model involves longer random

[24] http://ranksys.org/

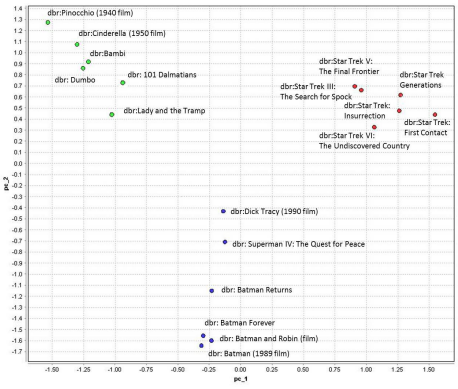



Figure 11.2: Two-dimensional PCA projection of the 200-dimensional Skip-Gram vectors of movies in Table 11.10.

walks, i.e., with depth 8. Comparing the LOD datasets, it clearly emerges that DBpedia lets to gain higher values than Wikidata for each metric involved, but it turns out that Wikidata is quite effective on `LibraryThing`, where the skip-gram vectors with depth of 4 exceed the corresponding DBpedia vectors. Moving to the features extracted from direct relations, the contribution of the "categories" stands clearly out, together with relations-values "rel-vals", especially when just incoming relations are considered: these features allow to achieve better results than the approaches based on translating embeddings, i.e., DB_TransE, DB_TransH and DB_TransR. The use of methods based on kernels for features extraction, i.e., WC_4 and WL_2_2 approaches, seems not to provide significant advantages to the recommendation algorithm.

To point out that the latent features built upon RDF graph are able to capture its structure, placing closely semantically similar items, some examples of the neighbouring sets retrieved using the graph embeddings technique are provided. These sets are directly exploited by the ItemKNN algorithm to produce recommendations. Table 11.10 is related to movies and to the strategy "DB2vec SG 500w 200v 4d", and displays that neighboring items are highly relevant and close to the query item, i.e., the item for which neighbors are searched for. Figure 11.2 depicts the 2D PCA projection of the movies in that table, showing that similar movies are actually projected closely to each other.

### 11.4.3 Conclusion

In this section we have shown that a content-based RS relying on the similarity between items computed according to our latent features vectors, outperform the same type of system using explicit features (e.g. types, categories,...) or features generated with the use of kernels, or related knoledge graph embedding systems, from both perspectives of accuracy and aggregate diversity.

This work has been continued in [258], where we developed a hybrid approach

based on factorization machines, using the RDF2vec embeddings as features. The results show that our approach not only outperforms state-of-the-art content-based approaches, but also state-of-the-art collaborative and hybrid recommender systems.

# Chapter 12

# Entity and Document Modeling using Semantic Web Graph Embeddings

Calculating entity relatedness and similarity are fundamental problems in numerous tasks in information retrieval, natural language processing, and Web-based knowledge extraction. While similarity only considers subsumption relations to assess how two objects are alike, relatedness takes into account a broader range of relations, i.e., the notion of relatedness is wider than that of similarity. For example, "Facebook" and "Google" are both entities of the class company, and they have high similarity and relatedness score. On the other hand, "Facebook" and "Mark Zuckerberg" are not similar at all, but are highly related, while "Google" and "Mark Zuckerberg" are not similar at all, and have somehow lower relatedness value.

## 12.1 Related Work

Both for entity and document ranking, as well as for the subtask of computing the similarity or relatedness of entities and documents, different methods using LOD have been proposed.

### 12.1.1 Entity Relatedness

Semantic relatedness of entities has been heavily researched over the past couple of decades. There are two main direction of studies. The first are approaches based on word distributions, which model entities as multi-dimensional vectors that are computed based on distributional semantics techniques [3, 95, 121]. The second are graph-based approaches relying on a graph structured knowledge base, or knowledge graph, which are the focus of this chapter.

Schuhmacher et al. [270] proposed one of the first approaches for entity ranking using the DBpedia knowledge graph. They use several path and graph based approaches for weighting the relations between entities, which are later used to calculate the entity relatedness. A similar approach is developed by Hulpus et al. [130], which uses local graph measures, targeted to the specific pair, while the previous approach uses global measures. More precisely, the authors propose the exclusivity-based relatedness measure that gives higher weights to relations that are less used in the graph. In [67] the authors propose a hybrid approach that exploits both textual and RDF data to rank resources in DBpedia related to the IT domain.

### 12.1.2 Entity and Document Similarity

As for the entity relatedness approaches, there are two main directions of research in the field of semantic document similarity, i.e., approaches based on word distributions, and graph-based approaches. Some of the earliest approaches of the first category make use of standard techniques like bag-of-words models, but also more sophisticated approaches. Explicit Semantic Analysis (ESA) [95] represents text as a vector of relevant concepts. Each concept corresponds to a Wikipedia article mapped into a vector space using the TF-IDF measure on the article's text. Similarly, Salient Semantic Analysis (SSA) [110] uses hyperlinks within Wikipedia articles to other articles as vector features, instead of using the full body of text.

Nunes et al. [202] present a DBpedia based document similarity approach, in which they compute a document connectivity score based on document annotations, using measures from social network theory. Thiagarajan et al. [297] present a general framework showing how spreading activation can be used on semantic networks to determine similarity of groups of entities. They experiment with Wordnet and the Wikipedia Ontology as knowledge bases and determine similarity of generated user profiles based on a 1-1 annotation matching.

Schumacher et al. [270] use the same measure used for entity ranking (see above) to calculate semantic document similarity. Similarly, Paul et al. [218] present an approach for efficient semantic similarity computation that exploits hierarchical and transverse relations in the graph.

One approach that does not belong to these two main directions of research is the machine-learning approach by Huang et al. [125]. The approach proposes a measure that assesses similarity at both the lexical and semantic levels, and learns from human judgments how to combine them by using machine-learning techniques.

Our work is, to the best of our knowledge, the first to exploit the graph structure using neural language modeling for the purpose of entity relatedness and similarity.

## 12.2 Approach

In this section, we introduce several approaches for entity and document modeling based on the previously built RDF2vec latent feature vectors for entities, presented in Chapter 8.

### 12.2.1 Entity Similarity

As previously mentioned, in the RDF2vec feature embedding space (see Section 13.3), semantically similar entities appear close to each other in the feature space. Therefore, the problem of calculating the similarity between two instances is a matter of calculating the distance between two instances in the given feature space. To do so, we use the standard cosine similarity measure, which is applied on the vectors of the entities. Formally, the similarity between two entities $e_1$ and $e_2$, with vectors $V_1$ and $V_2$, is calculated as the cosine similarity between the vectors $V_1$ and $V_2$:

$$sim(e_1, e_2) = \frac{V_1 \cdot V_2}{||V_1|| \cdot ||V_2||} \tag{12.1}$$

### 12.2.2 Document Similarity

We use those entity similarity scores in the task of calculating semantic document similarity. We follow a similar approach as the one presented in [218], where two documents are considered to be similar if many entities of the one document are similar to at least one entity in the other document. More precisely, we try to identify the most similar pairs of entities in both documents, ignoring the similarity of all other pairs.

Given two documents $d_1$ and $d_2$, the similarity between the documents $sim(d_1, d_2)$ is calculated as follows:

1. Extract the sets of entities $E_1$ and $E_2$ in the documents $d_1$ and $d_2$.
2. Calculate the similarity score $sim(e_{1i}, e_{2j})$ for each pair of entities in document $d_1$ and $d_2$, where $e_{1i} \in E_1$ and $e_{2j} \in E_2$
3. For each entity $e_{1i}$ in $d_1$ identify the maximum similarity to an entity in $d_2$ $max\_sim(e_{1i}, e_{2j} \in E_2)$, and vice versa.
4. Calculate the similarity score between the documents $d_1$ and $d_2$ as:

$$sim(d_1, d_2) = \frac{\sum_{i=1}^{|E_1|} max\_sim(e_{1i}, e_{2j} \in E_2) + \sum_{j=1}^{|E_2|} max\_sim(e_{2j}, e_{1i} \in E_1)}{|E_1| + |E_2|} \tag{12.2}$$

### 12.2.3 Entity Relatedness

In this approach we assume that two entities are related if they often appear in the same context. For example, “Facebook” and “Mark Zuckerberg”, which are highly related, are often used in the same context in many sentences. To calculate the

Table 12.1: Similarity-based relatedness Spearman's rank correlation results

| Model | IT companies | Hollywood Celebrities | Television Series | Video Games | Chuck Norris | All 21 entities |
|---|---|---|---|---|---|---|
| DB2vec SG 200w 200v 4d | 0.525 | 0.505 | 0.532 | 0.571 | 0.439 | 0.529 |
| DB2vec CBOW 200w 200v | 0.330 | 0.294 | 0.462 | 0.399 | 0.179 | 0.362 |
| DB2vec CBOW 500w 200v 4d | 0.538 | 0.560 | 0.572 | 0.596 | 0.500 | 0.564 |
| DB2vec CBOW 500w 500v 4d | 0.546 | 0.544 | 0.564 | 0.606 | 0.496 | 0.562 |
| DB2vec SG 500w 200v 4d | 0.508 | 0.546 | 0.497 | 0.634 | 0.570 | 0.547 |
| DB2vec SG 500w 500v 4d | 0.507 | 0.538 | 0.505 | 0.611 | 0.588 | 0.542 |
| DB2vec CBOW 500w 200v 8d | 0.611 | 0.495 | 0.315 | 0.443 | 0.365 | 0.461 |
| DB2vec CBOW 500w 500v 8w | 0.486 | 0.507 | 0.285 | 0.440 | 0.470 | 0.432 |
| DB2vec SG 500w 200v 8w | 0.739 | 0.723 | 0.526 | 0.659 | 0.625 | 0.660 |
| DB2vec SG 500w 500v 8w | **0.743** | **0.734** | **0.635** | **0.669** | **0.628** | **0.692** |
| WD2vec CBOW 200w 200v 4d | 0.246 | 0.418 | 0.156 | 0.374 | 0.409 | 0.304 |
| WD2vec CBOW 200w 500v 4d | 0.190 | 0.403 | 0.103 | 0.106 | 0.150 | 0.198 |
| WD2vec SG 200w 200v 4d | 0.502 | 0.604 | 0.405 | 0.578 | 0.279 | 0.510 |
| WD2vec SG 200w 500v 4d | 0.464 | 0.562 | 0.313 | 0.465 | 0.168 | 0.437 |
| Wiki2vec | 0.613 | 0.544 | 0.334 | 0.618 | 0.436 | 0.523 |

probability of two entities being in the same context, we make use of the RDF2Vec models and the set of sequences of entities generated as described in Section 13.3. Given a RDF2vec model and a set of sequences of entities, we calculate the relatedness between two entities $e_1$ and $e_2$, as the probability $p(e_1|e_2)$ calculated using the softmax function. In the case of a CBOW model, the probability is calculated as:

$$p(e_1|e_2) = \frac{exp(v_{e_2}^T v'_{e_1})}{\sum_{e=1}^{V} exp(v_{e_2}^T v'_e)}, \tag{12.3}$$

where $v'_e$ is the output vector of the entity $e$, and $V$ is the complete vocabulary of entities.

In the case of a skip-gram model, the probability is calculated as:

$$p(e_1|e_2) = \frac{exp(v'^T_{e_1} v_{e_2})}{\sum_{e=1}^{V} exp(v'^T_e v_{e_2})}, \tag{12.4}$$

where $v_e$ and $v'_e$ are the input and the output vector of the entity $e$, and $V$ is the complete vocabulary of entities.

## 12.3 Evaluation

For both tasks of determining entity relatedness and document similarity, benchmark datasets exist. We use those datasets to compare the use of RDF2Vec models against state of the art approaches.

Table 12.2: Context-based relatedness Spearman's rank correlation results

| Model | IT companies | Hollywood Celebrities | Television Series | Video Games | Chuck Norris | All 21 entities |
|---|---|---|---|---|---|---|
| DB2vec SG 200w 200v 4d | 0.643 | 0.547 | 0.583 | 0.428 | 0.591 | 0.552 |
| DB2vec CBOW 200w 200v | 0.361 | 0.326 | 0.467 | 0.426 | 0.208 | 0.386 |
| DB2vec CBOW 500w 200v 4d | 0.671 | 0.566 | 0.591 | 0.434 | 0.609 | 0.568 |
| DB2vec CBOW 500w 500v 4d | 0.672 | 0.622 | 0.578 | 0.440 | 0.581 | 0.578 |
| DB2vec SG 500w 200v 4d | 0.666 | 0.449 | 0.611 | 0.360 | 0.630 | 0.526 |
| DB2vec SG 500w 500v 4d | 0.667 | 0.444 | 0.609 | 0.389 | 0.668 | 0.534 |
| DB2vec CBOW 500w 200v 8d | 0.579 | 0.484 | 0.368 | 0.460 | 0.412 | 0.470 |
| DB2vec CBOW 500w 500v 8d | 0.552 | 0.522 | 0.302 | 0.487 | 0.665 | 0.475 |
| DB2vec SG 500w 200v 8d | **0.811** | **0.778** | **0.711** | **0.658** | **0.670** | **0.736** |
| DB2vec SG 500w 500v 8d | 0.748 | 0.729 | 0.689 | 0.537 | 0.625 | 0.673 |
| WD2vec CBOW 200w 200v 4d | 0.287 | 0.241 | -0.025 | 0.311 | 0.226 | 0.205 |
| WD2vec CBOW 200w 500v 4d | 0.166 | 0.215 | 0.233 | 0.335 | 0.344 | 0.243 |
| WD2vec SG 200w 200v 4d | 0.574 | 0.671 | 0.504 | 0.410 | 0.079 | 0.518 |
| WD2vec SG 200w 500v 4d | 0.661 | 0.639 | 0.537 | 0.395 | 0.474 | 0.554 |
| Wiki2vec | 0.291 | 0.296 | 0.406 | 0.353 | 0.175 | 0.329 |

### 12.3.1 Entity Relatedness

For evaluating the entity relatedness approach, we use the KORE dataset [121]. The dataset consists of 21 main entities, whose relatedness to the other 20 entities each has been manually assessed, leading to 420 rated entity pairs. We use the Spearman's rank correlation as an evaluation metric.

We use two approaches for calculating the relatedness rank between the entities, i.e. (i) the entity similarity approach described in section 12.2.1; (ii) the entity relatedness approach described in section 12.2.3.

We evaluate each of the RDF2Vec models separately. Furthermore, we also compare to the Wiki2vec model[1], which is built on the complete Wikipedia corpus, and provides vectors for each DBpedia entity.

Table 12.1 shows the Spearman's rank correlation results when using the entity similarity approach. Table 12.2 shows the results for the relatedness approach. The results show that the DBpedia models outperform the Wikidata models. Increasing the number of walks per entity improves the results. Also, the skip-gram models outperform the CBOW models continuously. We can observe that the relatedness approach outperforms the similarity approach.

Furthermore, we compare our approaches to several state-of-the-art graph-based entity relatedness approaches:

- baseline: computes entity relatedness as a function of distance between the entities in the network, as described in [270].

- KORE: calculates keyphrase overlap relatedness, as described in the original KORE paper [121].

[1] `https://github.com/idio/wiki2vec`

Table 12.3: Spearman's rank correlation results comparison to related work

| Approach | IT companies | Hollywood Celebrities | Television Series | Video Games | Chuck Norris | All 21 entities |
|---|---|---|---|---|---|---|
| baseline | 0.559 | 0.639 | 0.529 | 0.451 | 0.458 | 0.541 |
| KORE | 0.759 | 0.715 | 0.599 | **0.760** | 0.498 | 0.698 |
| CombIC | 0.644 | 0.690 | 0.643 | 0.532 | 0.558 | 0.624 |
| ER | 0.727 | 0.643 | 0.633 | 0.519 | 0.477 | 0.630 |
| DB_TransE | -0.023 | 0.120 | -0.084 | 0.353 | -0.347 | 0.070 |
| DB_TransH | -0.134 | 0.185 | -0.097 | 0.204 | -0.044 | 0.035 |
| DB_TransR | -0.217 | 0.062 | 0.002 | -0.126 | 0.166 | 0.058 |
| DB2Vec Similarity | 0.743 | 0.734 | 0.635 | 0.669 | 0.628 | 0.692 |
| DB2Vec Relatedness | **0.811** | **0.778** | **0.711** | 0.658 | **0.670** | **0.736** |

- CombIC: semantic similarity using a Graph Edit Distance based measure [270].
- ER: Exclusivity-based relatedness [130].

The comparison shows that our entity relatedness approach outperforms all the rest for each category of entities. Interestingly enough, the entity *similarity* approach, although addressing a different task, also outperforms the majority of state of the art approaches.

### 12.3.2 Document Similarity

To evaluate the document similarity approach, we use the LP50 dataset [160], namely a collection of 50 news articles from the Australian Broadcasting Corporation (ABC), which were pairwise annotated with similarity rating on a Likert scale from 1 (very different) to 5 (very similar) by 8 to 12 different human annotators. To obtain the final similarity judgments, the scores of all annotators are averaged. As a evaluation metrics we use Pearson's linear correlation coefficient and Spearman's rank correlation plus their harmonic mean.

Again, we first evaluate each of the RDF2Vec models separately. Table 12.4 shows document similarity results. As for the entity relatedness, the results show that the skip-gram models built on DBpedia with 8 hops lead to the best performances.

Furthermore, we compare our approach to several state-of-the-art graph-based document similarity approaches:

- TF-IDF: Distributional baseline algorithm.
- AnnOv: Similarity score based on annotation overlap that corresponds to traversal entity similarity with radius 0, as described in [218].
- Explicit Semantic Analysis (ESA) [95].
- GED: semantic similarity using a Graph Edit Distance based measure [270].

Table 12.4: Document similarity results - Pearson's linear correlation coefficient ($r$), Spearman's rank correlation ($\rho$) and their harmonic mean $\mu$

| Model | $r$ | $\rho$ | $\mu$ |
|---|---|---|---|
| DB2vec SG 200w 200v 4d | 0.608 | 0.448 | 0.516 |
| DB2vec CBOW 200w 200v 4d | 0.562 | 0.480 | 0.518 |
| DB2vec CBOW 500w 200v 4d | 0.681 | 0.535 | 0.599 |
| DB2vec CBOW 500w 500v 4d | 0.677 | 0.530 | 0.594 |
| DB2vec SG 500w 200v 4d | 0.639 | 0.520 | 0.573 |
| DB2vec SG 500w 500v 4d | 0.641 | 0.516 | 0.572 |
| DB2vec CBOW 500w 200v 8d | 0.658 | 0.491 | 0.562 |
| DB2vec CBOW 500w 500v 8d | 0.683 | 0.512 | 0.586 |
| DB2vec SG 500w 200v 8d | **0.708** | **0.556** | **0.623** |
| DB2vec SG 500w 500v 8d | 0.686 | 0.527 | 0.596 |
| WD2vec CBOW 200w 200v 4d | 0.568 | 0.383 | 0.458 |
| WD2vec CBOW 200w 500v 4d | 0.593 | 0.386 | 0.467 |
| WD2vec SG 200w 200v 4d | 0.606 | 0.385 | 0.471 |
| WD2vec SG 200w 500v 4d | 0.613 | 0.343 | 0.440 |
| Wiki2vec | 0.662 | 0.513 | 0.578 |
| DB_TransE | 0.565 | 0.432 | 0.490 |
| DB_TransH | 0.570 | 0.452 | 0.504 |
| DB_TransR | 0.578 | 0.461 | 0.513 |

Table 12.5: Comparison of the document similarity approach to the related work

| Approach | $r$ | $\rho$ | $\mu$ |
|---|---|---|---|
| TF-IDF | 0.398 | 0.224 | 0.287 |
| AnnOv | 0.590 | 0.460 | 0.517 |
| LSA | 0.696 | 0.463 | 0.556 |
| SSA | 0.684 | 0.488 | 0.570 |
| GED | 0.630 | \ | \ |
| ESA | 0.656 | 0.510 | 0.574 |
| GBSS | 0.704 | 0.519 | 0.598 |
| DB2Vec | **0.708** | **0.556** | **0.623** |

- Salient Semantic Analysis (SSA), Latent Semantic Analysis (LSA) [110].
- Graph-based Semantic Similarity (GBSS) [218].

The results for the related approaches were copied from the respective papers, except for ESA, which was copied from [218], where it is calculated via public ESA REST endpoint[2]. The results show that our document similarity approach outperforms all of the related approaches for both Pearson's linear correlation coefficient and Spearman's rank correlation, as well as their harmonic mean.

We do not compare our approach to the machine-learning approach proposed by Huang et al. [125], because that approach is a supervised one, which is tailored towards the dataset, whereas ours (as well as the others we compare to) are unsupervised.

[2] `http://vmdeb20.deri.ie:8890/esaservice`

# Chapter 13

# Taxonomy Induction Using Knowledge Graph Embeddings

As shown in chapter 3, semantic ontologies and hierarchies are established tools to represent domain-specific knowledge with dozens of scientific, industrial and social applications [99] and, in knowledge bases, the natural way to structure the knowledge. A basic building block of an ontology is a class, where the classes are organized with "is-a" relations, or class subsumption axioms (e.g., each *city* is a *place*). It is crucial to define a high quality class hierarchy for a knowledge base in order to allow effective access to the knowledge base from various Natural Language Processing, Information Retrieval, and any Artificial Intelligence systems and tools.

However, manually curating a class hierarchy for a given knowledge graph is time consuming and requires a high cost. For example the DBpedia Ontology [162], which is a central hub for many applications in the Semantic Web domain [267], has been manually created based on the most commonly used infoboxes within Wikipedia.

**The work presented in this chapter has been published before as: "Petar Ristoski, Stefano Faralli, Simone Paolo Ponzetto, Heiko Paulheim: *Large-scale taxonomy induction using entity and word embeddings.* Proceedings of the International Conference on Web Intelligence, pages 81–87, 2017" [248]**

## 13.1 Introduction

Many recent studies propose automatic extraction of class hierarchies [104, 153, 316]. The importance of automatic approaches for the induction class hierarchy becomes more apparent when we deal with large scale automatically acquired knowledge bases such as the WebIsA database (WebIsADb) [274].

The WebIsADb is a large collection of more than 400 million hypernymy re-

lations. Relations are extracted from the CommonCrawl,[1] by means of Hearst-like patterns [113]. Being extracted from raw text from the very diverse Web sources, and using heuristics of varying reliability, the WebIsaDB provides a good coverage, but rather low precision, and cannot be applied as a taxonomy as is.

In recent years, word embedding models have been used heavily in many NLP applications. Such approaches take advantage of the word order in text documents, explicitly modeling the assumption that closer words in the word sequence are statistically more dependent. In the resulting semantic vector space, similar words appear close to each other, and simple arithmetic operations can be executed on the resulting vectors. One of the widely used approach is the word2vec neural language model [177, 178]. Word2vec is a particularly computationally-efficient two-layer neural net model for learning word embeddings from raw text. Another widely used approach is GloVe [232], which in comparison to word2vec is not a predictive model, but a count-based model, which learns word vectors by doing dimensionality reduction on a co-occurrence counts matrix. The studies suggest that both models show comparable performances on many NLP tasks [164].

In this chapter, we present the *TIEmb* approach for automatic unsupervised class subsumption axiom extraction from knowledge bases using entity and text embeddings. More precisely, we use the RDF2vec embeddings, presented in chapter 8. The underlying assumptions behind our approach are: (i) the majority of all instances of the same class are positioned close to each other in the embedded space; (ii) each class in the knowledge base can be represented as a cluster of the instances in the embedded space, defined with a centroid and an average radius; (iii) clusters that completely or partially subsume each other, indicate class subsumption axiom.

Figure 13.1 shows an example of three class clusters projected into a two dimensional feature space. Each of the classes is represented with the class instances, centroid and a radius. As we can observe, the centroids of the "Football Player" and "Basketball Player" classes are within the radius of the "Athlete" class, which indicates that the "Football Player" and "Basketball Player" are subclasses of the "Athlete" class.

The contributions of this chapter are the following ones:

- We present a novel unsupervised approach to induce class subsumption axioms using entity and word embeddings.
- We show that such approach can be applied on large knowledge bases, like DBpedia and the WebIsADb.
- Finally, we provide the resulting class subsumption axioms in the *Person* and *Place* domains extracted from the WebIsADb. This goes in the direction of semantifying the WebIsADb, and the CommonCrawl in general.

[1] `http://commoncrawl.org/`

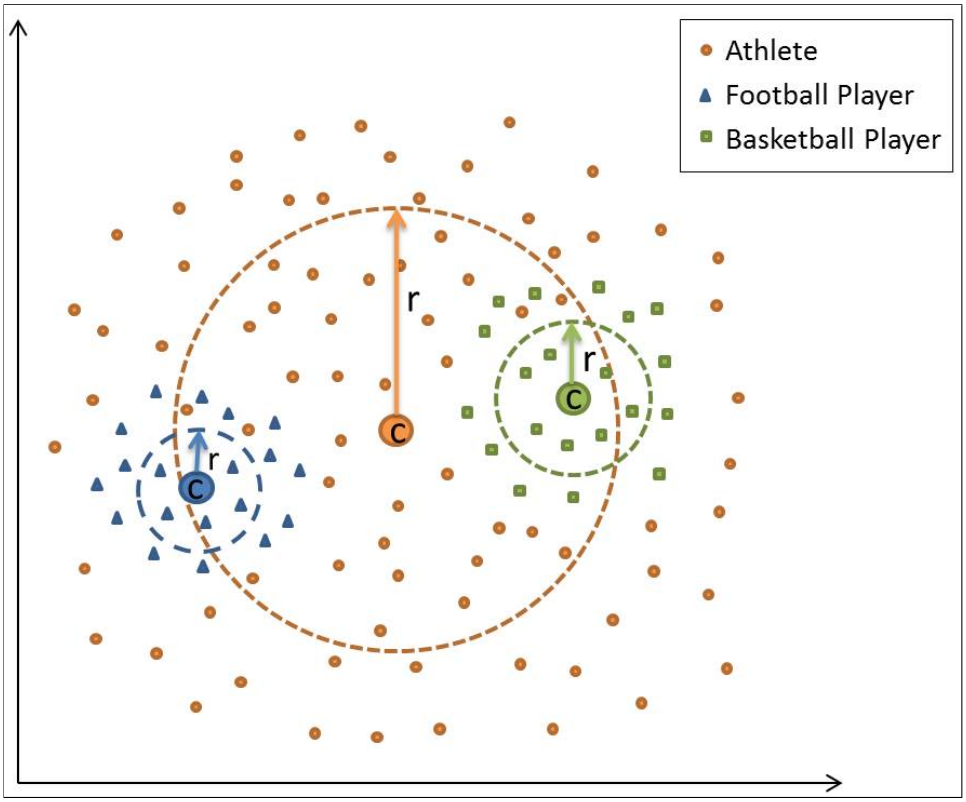



Figure 13.1: Example of three classes in two-dimensional feature space

## 13.2 Related Work

The backbone of ontologies typically consist of hierarchy of hypernymy relations, namely "IsA" relations, typically pairs of the kind $(t, h)$ where $t$ is a concept/term and $h$ one of its generalizations. Hence, in the construction of knowledge bases, the induction of taxonomies represents an intermediate fundamental step. However, manually constructing taxonomies is a very demanding task, requiring a large amount of time and effort. A quite recent challenge, referred to as ontology learning, consists of automatically or semi-automatically creating a lexicalized ontology using textual data from corpora or the Web (see [18, 287] for a survey on the task). As a result, the heavy requirements of manual ontology construction are drastically reduced.

The task of lexicalized taxonomy induction can be started with the extraction of domain specific hypernymy relations from texts. To this end, [153] use Hearst-like patterns [113] to bootstrap the extraction of terminological sisters terms and hypernyms. Instead, in [316] the extraction of hypernymy relations is performed with a classifier, which is trained on a set of manually annotated definitions from Wikipedia, being able to detect definitional sentences and to extract the definiendum and the hypernym. In the above mentioned systems, the harvested hypernymy

relations are then arranged into a taxonomy structure.

In general, all such lexical-based approaches suffer from the limitation of not being sense-aware, which results in spurious taxonomic structures. To cope with such limitations, in [82] the authors adopt a corpus-based unsupervised distributional semantics method to harvest fully disambiguated sense inventories, as well as a new approach to clean distributional semantics-based acquired knowledge graphs [83].

In all the above web-based approaches, a problem arises when the systems open to the Web. In fact, in the last year the majority of the Web search engines do not allow to programmatically query the Web. The WebIsADb [274] addresses the "unavailability" of the Web indices, providing a large database (more than 400M tuples) of hypernymy relations extracted from the CommonCrawl web corpus.[2]

Another group of approaches - instead of inducing a taxonomy from scratch - involve statistical evidence to induce structured hierarchies on existing data sources. In [318] (among others) a schema is statistically induced from the large amount of RDF triples on the Web. Enabling suitable schemas for all those application where logical inference is required.

Recently, word embeddings representations are involved in the task of knowledge hierarchical organization [88], [330], [104]. However, these methods are only considering hypernym-hyponym relationship extraction between lexical terms, using word embeddings. Instead, in our approach we focus on extracting class subsumption axioms, where each class is represented with a set instances.

## 13.3 Approach

Our approach makes use of vector space embeddings. Those are projections of each instance – e.g., a word or an entity in a graph – into a low dimensional vector space. The core assumption that we exploit in our approach is that two similar entities are positioned close to each other in that vector space.

Following that intuition, we can assume that entities which belong to the same class are positioned close to each other in the vector space, since they share some commonalities. Furthermore, instances of a more specific class should be positioned closer to each other on average than instances of a broader class. For example, the class of basketball players is more specific than the class of athletes, so that we assume that basketball players are on average closer to each other in the vector space than athletes.

Furthermore, since basketball players are a subclass of athletes, we assume that the majority of vector space points corresponding to basketball players will be positioned inside the cluster formed by the athletes in general, hence, their centroids will be close to each other as well.

The TIEmb approach builds upon those two assumptions. Its pseudocode is shown in Algorithm 3. The algorithm has two inputs: (1) a knowledge base as

[2] `https://commoncrawl.org`

Table 13.1: Results for class subsumption axioms extraction using DBpedia ontology as a gold standard

| Method | Precision | Recall | F-score |
|---|---|---|---|
| rel out | 0.056 | 0.076 | 0.064 |
| rel in | 0.097 | 0.209 | 0.132 |
| rel in & out | 0.104 | 0.219 | 0.141 |
| TIEmb | **0.594** | **0.465** | **0.521** |

its input, which contains a set of instances, where each instance has one or more class types, and (2) the knowledge base embeddings, where each instance is represented as n-dimensional feature vector. The output of the algorithm is a set of class subsumption axioms.

In the first step, the algorithm calculates the centroid and radius for all the classes in the knowledge base (lines 2-8). To do so, we select all instances for all the classes, where each instance is represented with the corresponding embedding vector $v_i$. The centroid of a class represents a vector, which is calculated as the average of the vectors of all the instances that belong to the class (line 6). The radius of the class is calculated as the average distance of all instances in the class to the centroid (line 7).

In the next step, the class subsumption axioms are extracted (lines 9-24). First, for each class $c_1$ in the knowledge base we generate a set of candidate axioms. The algorithm iterates over all pairs of classes, and checks for two conditions before it creates new candidate axiom: for the classes $c_1$ and $c_2$, if the distance between the centroids of the classes, $distance_{c_1c_2}$, is smaller than the radius of $c_2$ and the radius of $c_1$ is smaller than the radius of $c_2$, then a new candidate axiom is generated where $c_1$ is a subclass of $c_2$, i.e., $c_1 \sqsubseteq c_2$. Each candidate axiom together with the distance $distance_{c_1c_2}$, are stored in a list. As a final axiom, we select the axiom with the smallest distance.

In the final step, the algorithm computes the transitive closure over all the previously extracted axioms (lines 25-44). We also make sure that there are no cycles in the final class hierarchy (line 34).

## 13.4 Experiments

We perform two experiments: (i) applying the proposed approach on the DBpedia dataset using the DBpedia ontology as a gold standard (see Section 13.4.1); (ii) applying the proposed approach on the WebIsA database in unsupervised manner (see Section 13.4).

**Algorithm 3** Algorithm for taxonomy induction from a knowledge base

**Data:** $KB$: Knowledge base, $V_{KB}$: Knowledge base embeddings
**Result:** $A$: Set of class subsumption axioms

$A := \emptyset$
# Calculate the centroid and radius for each class in the knowledge base
$C_{KB} = \{c \mid \exists i \; typeOf \; c \wedge i \in KB\}$
**foreach** *class* $c \in C_{KB}$ **do**
  $I_c = \{i \mid i \; typeOf \; c \wedge i \in KB \wedge \; \exists v_i \in V_{KB}\}$
  $c.centroid = \frac{1}{|I_c|} \sum_{i \in I_c} v_i$
  $c.radius = \sqrt{\frac{1}{|I_c|} \sum_{i \in I_c} (v_i - c.centroid)^2}$
**end**
# Extract class subsumption axioms
**foreach** *class* $c_1 \in C$ **do**
  $A_{c_1} := \emptyset$
  **foreach** *class* $c_2 \in C$ **do**
    **if** $c_1 == c_2$ **then**
      continue
    **end**
    $distance_{c_1 c_2} = distance(c_1.centroid, \; c_2.centroid)$
    **if** $distance_{c_1 c_2} \leq c_2.radius \; and \; c_1.radius < c_2.radius$ **then**
      $axiom = c_1 \sqsubseteq c_2$
      add $[axiom, distance]$ to $A_{c_1}$
    **end**
  **end**
  sort $A_{c_1}$ in ascending order
  add $A_{c_1}.first()$ to $A$
**end**
# Compute transitive closure
$change = true$
**while** $change == true$ **do**
  **foreach** *axiom* $a \in A$ **do**
    $change = false$
    $subClass = axiom.getSubClass()$
    $superClass = axiom.getSuperClass()$
    $superClassesAxioms = \{c, axiom \mid \exists axiom \in A \wedge axiom = superClass \sqsubseteq c\}$
    **foreach** *classAxiom* $s \in superClassesAxioms$ **do**
      **if** $s.class == subClass$ **then**
        remove $s.axiom$ from $A$
        continue
      **end**
      $axiom = subClass \sqsubseteq s$
      **if** $axiom \notin A$ **then**
        add $axiom$ to $A$
        $change = true$
      **end**
    **end**
  **end**
**end**
return $A$

Table 13.2: Results for the class subsumption axiom induction in the *Person* domain

| Method | DBpedia Coverage | Extra Coverage(%) | Precision (random 100) |
|---|---|---|---|
| Association Rules | **0.44** | **31301.63** | 0.25 |
| RDF2Vec TIEmb | 0.31 | 3594.44 | **0.42** |
| GloVe TIEmb | 0.29 | 1520.00 | 0.29 |

Table 13.3: Results for the class subsumption axiom induction in the *Place* domain

| Method | DBpedia Coverage | Extra Coverage(%) | Precision (random 100) |
|---|---|---|---|
| Association Rules | **0.25** | **11029.71** | 0.45 |
| RDF2Vec TIEmb | 0.21 | 3808.57 | **0.54** |
| GloVe TIEmb | 0.17 | 301.14 | 0.48 |

### 13.4.1 Embedding the DBpedia Ontology

To evaluate the approach we use the RDF2vec graph embeddings (described in Chapter 8), built on the DBpedia knowledge graph.

We compare the embedding vectors to three baseline feature generation approaches, as proposed in [225]. We generate features derived from generic relations in the graph, i.e., we generate a feature for each incoming (rel in) or outgoing relation (rel out) of an entity, ignoring the value or target entity of the relation. Furthermore, we combine both incoming and outgoing relations (rel in & out).

The results are shown in Table 13.1. The results show that we are able to identify 46.5% of all the class subsumption axioms, and from the ones we discovered, 59.4% exist in DBpedia. Furthermore, the results show that using entity embeddings significantly outperforms the baseline feature generation approaches.

The error analysis showed that the algorithm often extracts subsumption axioms for classes on the same level in the hierarchy, or siblings classes, e.g. *dbo:Bird* $\subseteq$ *dbo:Mammal*. The reason for such false positives is that the centroids of the classes are positioned very close to each other in the embeddings space. Furthermore, some of the false positives would not necessarily be incorrect, but those axioms simply do not exist in the DBpedia ontology, e.g., *dbo:Senator* $\subseteq$ *dbo:OfficeHolder*. Since the DBpedia ontology, like all data sets on the Semantic Web, follows the open world assumption, 59.4% are only a pessimistic estimate for our approach's precision (see [223] for a discussion).

As a second comparison, we tried to compare our approach to the approach proposed by Völker and Niepert [318], who propose to use association rule mining for deriving subsumption axioms. The core idea of their approach is that if instances of class A are often also of class B, then A should be a subclass of B. However, the approach was not able to generate any axioms on the dataset we used, since in this dataset, only the most specific class is assigned to each instance, hence, co-occurrences such as those which their approach tries to exploit are rarely found in the dataset.

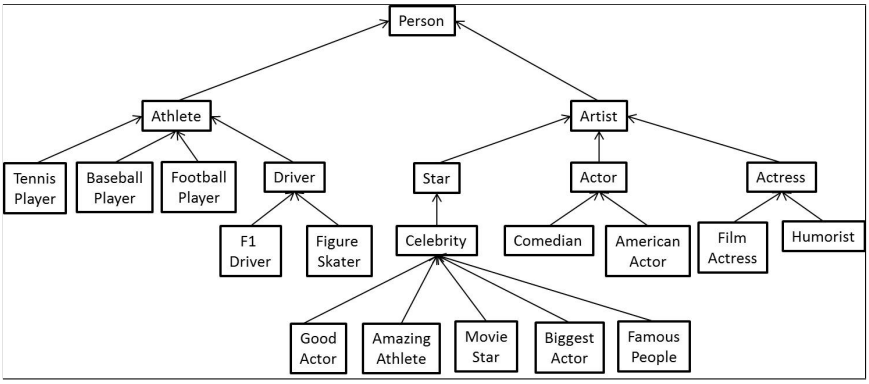



Figure 13.2: Excerpt of the *Person* top level hierarchy

### 13.4.2 Inducing Ontologies from the WebIsADb

The WebIsA database (WebIsADb) is a database of word pairs which reflect subsumption relations. It was generated from the CommonCrawl by applying a set of patterns like Hearst patterns [113], e.g., *X is a Y* or *Y such as X* to the large corpus of text in the common crawl. Given that the input is raw text from the Web, and all the patterns are heuristic, the corpus is assumed to provide high coverage, but low precision. Thus, simply arranging all the axioms from the WebIsADb into a graph would not constitute a taxonomy, but rather a densely connected graph with many cycles.

In the second set of experiments, we apply the proposed approach on the WebIsADb in order to extract meaningful class hierarchies for the underlying data. As the WebIsADb contains $120,992,248$ entities, extracting a single class hierarchy for the complete database is not a trivial task. Therefore, we narrowed the experiments to extract 2 domain specific class hierarchies, i.e., *Person* and *Place*.

To extract the class hierarchy, first we need to select the domain specific instances from the WebIsADb and identify a finite set of classes for which we need to extract class subsumption axioms. To do so, we use domain specific classes from DBpedia as filters. First, we select all the subclasses of dbo:Person (184 in total) and dbo:Place (176 in total) in DBpedia, and use them as the initial set of classes for each domain separately. Then, we select all the instances of these classes in WebIsADb. To identify the rest of the domain specific classes in WebIsADb, we expand the set of classes by adding all siblings of the corresponding class within the WebIsADb. For example, we use dbo:SoccerPlayer to select all the instances of type “soccer player” in WebIsADb, e.g., “Cristiano Ronaldo” is such an instance. In the next step, we expand the initial set of classes with all the classes assigned to the “soccer player” instances, e.g., from the instance “Cristiano Ronaldo”, we will add the following classes in the set of *Person* classes: “Player”, “Portuguese Footballer”, “Star”, “Great Player”, etc.

Once we have defined the set of classes for each domain, we select all the

instances for each class, which represents the input knowledge base for Algorithm 3. We experiment with entity and word embeddings. As entity embeddings we use DBpedia RDF2vec embeddings, which are built similarly as described in the previous section, only this time we also use the instance transitive types. To link the WebIsADb instances to DBpedia, we use exact string matching.

For word embeddings, we use GloVe embeddings [232], trained on the complete Common Crawl.[3] The model provides 300-dimensional embedding vectors for 2.2 million tokens. In case of multiple tokens in the WebIsADb instance, the final vector is calculated as the average of the vectors of all the tokens in the instance.

Again, we compare our approach to the association rule mining-based approach proposed in [318]. To do so, for each instance, we generate a transaction of all the instance's types. Then, for each of the classes separately, we learn class subsumption axioms using the standard Apriori algorithm [4]. We consider all the rules with support and confidence value above 50.

To evaluate the induced class subsumption axioms, we use the DBpedia ontology as a reference class hierarchy, i.e., (i) we count how many of the class subsumption axioms defined in the DBpedia ontology we identified in WebIsADb (DBpedia coverage); (ii) we count how many more axioms were discovered compared to the DBpedia Ontology (Extra Coverage); (iii) we manually determine the precision on 100 randomly selected axioms from all of the extracted axioms.

First, we manually map each DBpedia class to the corresponding WebIsADb class. For the *Person* domain, we were able to map 0.99% of the classes, and there is 1466.67% extra class coverage for the same domain in WebIsADb. For the *Place* domain, we were able to map 0.71% of the classes, and there is 1161.93% extra class coverage for the same domain in WebIsADb. The results for the *Person* and the *Place* domain are shown in Table 13.2 and 13.3, respectively. We can observe that although the coverage of axioms is slightly lower using the graph embeddings and TIEmb, we are able to mine subsumption axioms at much higher precision.

The extracted class subsumption axioms can be found online.[4] Excerpts of the top levels of the *Person* and *Place* hierarchies are shown in Figure 13.2 and Figure 13.3, respectively.

## 13.5 Conclusion and Outlook

Taxonomies are an important backbone of formal knowledge representations. However, at large scale, they cannot be created manually with reasonable efforts, thus, automatic approaches have to be used. In this chapter, we have accordingly shown that using word and knowledge base embeddings are suitable approaches for inducing large scale taxonomies from knowledge graphs such as DBpedia or the

[3] `https://nlp.stanford.edu/projects/glove/`

[4] `http://data.dws.informatik.uni-mannheim.de/rdf2vec/TI/webisa/`

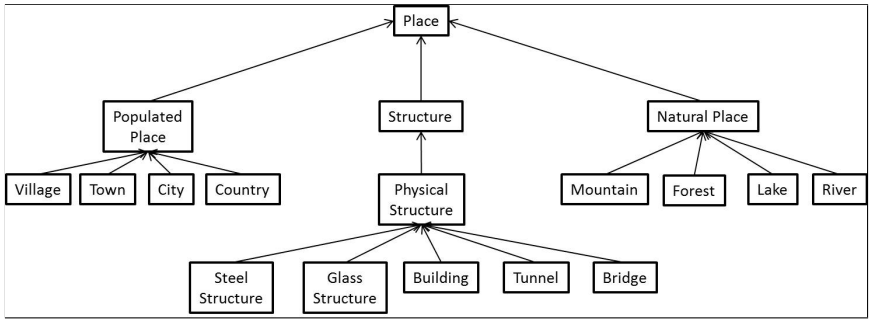



Figure 13.3: Excerpt of the *Place* top level hierarchy

WebIsADB. The approach relies on vector space embeddings of entities and exploits the proximity preserving properties of such embeddings approaches such as GloVe or RDF2vec.

Using WebIsADB, we were able to create hierarchies of thousands of classes at decent precision. We created two example hierarchies, i.e., persons and places, but the approach is capable of generating class hierarchies for any seed concept at hand. In general, the approach cannot only be used for taxonomy induction, but also for the problem of type prediction [224]. Here, again exploiting the proximity relations in the embedding space, each instance can be assigned to the types with the closest cluster centroid(s).

In the future, it will be interesting to see how embeddings coming from text and graphs can be combined reasonably. This will allow for even more concise induction of taxonomies. Furthermore, we want to investigate how higher-level semantic knowledge, such as class restrictions or complementarity, can be mined using an embedding-based approach. That way, using the Common Crawl as a representative sample of the knowledge that exists on the Web, we will be able to create large-scale semantic knowledge representations directly from Web data.

# Chapter 14

# Thesis Conclusion

This chapter summarizes the three previous parts of the thesis and outlines the major contributions of the thesis.

In Chapter 2, we showed that with the advent and growth of Linked Open Data, information from the Semantic Web can be used beneficially in the data mining and knowledge discovery process. However, the knowledge discovery process is still not tapping the full potential that is provided by the Semantic Web. In this thesis, we developed several approaches that exploit Semantic Web knowledge graphs in order to aid different steps of the knowledge discovery process.

## 14.1 PART I: Mining Semantic Web Knowledge Graphs

In Part I, we investigate the hypothesis for exploiting Semantic Web knowledge graphs as a useful source for background knowledge in data mining, which answers research question RS1 and RS2 from Section 1.1. In Chapter 4, we introduced a collection of datasets for benchmarking machine learning approaches for the Semantic Web. Such a collection of datasets can be used to conduct quantitative performance testing and systematic comparisons of approaches. We use this collection of datasets to evaluate the approaches introduced in this thesis. In Chapter 5, we describe a set of strategies for creating features from types and relations in Semantic Web knowledge graphs. We show that such simple graph transformation strategies can significantly improve the performances for a given data mining task on a given data set. As the number of generated features rapidly grows, in Chapter 6, we provide a strategy for feature selection in hierarchical feature space, in order to select only the most informative and most representative features for a given dataset. In the final chapter of this part, Chapter 7, we provide an end-to-end tool for mining the Web of Linked Data, which provides functionalities for each step of the knowledge discovery process, i.e., linking local data to a Semantic Web knowledge graph, integrating features from multiple knowledge graphs, feature generation and selection, and building machine learning models. Such a tool allows for bridging the gap between the Web of Data and data mining, and which

can be used for carrying out sophisticated analysis tasks on and with Linked Open Data.

## 14.2 PART II: Semantic Web Knowledge Graphs Embeddings

The feature generation strategies introduced in Part I do not scale when the input dataset is large, i.e., the number of generated features quickly becomes unmanageable. To address this problem, in Chapter 8 in Part II, we introduced an approach for Semantic Web knowledge graphs embedding, named RDF2Vec, which answers research question RS3 from Section 1.1. The approach uses language modeling approaches for unsupervised feature extraction from sequences of words, and adapts them to Semantic Web knowledge graphs. We generate sequences by leveraging local information from graph sub-structures, harvested by Weisfeiler-Lehman Subtree RDF Graph Kernels and graph walks, and learn latent numerical representations of entities in RDF graphs. Our evaluation shows that such vector representations outperform existing techniques for the propositionalization of RDF graphs on a variety of different predictive machine learning tasks, and that feature vector representations of general knowledge graphs such as DBpedia and Wikidata can be easily reused for different tasks.

In Chapter 9, we extend the RDF2Vec approach by examining methods to direct the random walks in more meaningful ways, i.e., being able to capture more important information about each entity in the graph. The chapter evaluates a dozen weighting schemes which influence the walks and, thus, the resulting sequences.

## 14.3 PART III: Applications of Semantic Web Knowledge Graphs

In Part III, we listed all the applications that use the approaches described in the previous two parts of the thesis, which answers research question RS4 from Section 1.1. In Chapter 10 we presented the Web-based tool *ViCoMap*, which allows automatic correlation analysis and visualizing statistical data on maps using Semantic Web knowledge graphs, which is based on the RapidMiner LOD extension (described in Chapter 7). In Chapter 11, we introduced three different content-based recommender systems that utilize Semantic Web knowledge graphs. The first approach is based on graph metrics, the second approach is based on a hybrid approach using flat features extracted from Semantic Web knowledge graphs, and the third approach uses graph embeddings (introduced in Chapter 8) as features to build a content-based recommender system. In Chapter 12, we show that the graph embeddings, introduced in Chapter 8, can be used for the task of entity and document modeling, which are fundamental problems in numerous tasks in information retrieval, natural language processing, and Web-based knowledge

extraction. In Chapter 13, we tackle the problem of taxonomy induction, which is crucial for maintaining high quality Semantic Web knowledge graphs. Thus, this chapter presents an approach that makes use of the graph embeddings introduced for automatic unsupervised class subsumption axiom extraction from large semi-structured knowledge bases.

Besides the applications described in this part of the thesis, there is a list of further applications where the approaches described in this thesis have been used. Furthermore, there are several third party applications that utilize some of the approaches described in this thesis.

The RapidMiner Linked Open Data extension (described in Chapter 7) has been used in several further use cases.

In [265], we aim at finding special characteristics of a given instance from a Semantic Web knowledge graph, given a contrast set. To that end, data about the instance at hand, as well as its contrast set, are retrieved from DBpedia. Attribute-wise outlier detection, which computes outlier scores for single attribute values [227], is exploited for identifying those attribute values of the instance that are significantly different from those of the other instances.

In [251], we use DBpedia, Eurostat, and LinkedGeoData to retrieve features to discover which factors correlate with the unemployment rate in French regions. Later, we used the corresponding links from DBpedia to GADM[1] to automatically retrieve the geographical shape data for each region, and visualize the findings on a map.

Furthemore, we have used the tool to implement baseline recommender system approaches, for the recommendation of movies [256] and music albums [257].

In [222], the LOD extension was used for the purpose of linkage error detection between LOD datasets. To that end, links between datasets are projected into a high dimensional feature space, and different multi-dimensional outlier detection techniques, taken from RapidMiner's *Anomaly Detection extension* [100], are used to identify wrong links. The approach was evaluated on two datasets of `owl:sameAs` links: DBpedia - Peel Sessions[2], and DBpedia - DBTropes[3]. The LOD extension was used to generate the feature vectors for each link, i.e., direct types, and all ingoing and outgoing properties. The evaluation showed promising results, with an area under the ROC curve up to 0.86 (i.e., wrong links get lower scores than correct links with a probability of 86%), and an F-measure up to 0.54.

In another use case, the extension is used to implement an approach that combines NLP techniques and background knowledge from DBpedia for finding disputed topics in news sites. To identify these topics, newspaper articles are annotated with DBpedia concepts, using the DBpedia Spotlight linking operator. For each article, a polarity score is computed. Then, the extracted DBpedia categories are used to identify those categories revealing significant deviations in polarity

[1] `http://gadm.geovocab.org/`
[2] `http://dbtune.org/bbc/peel/`
[3] `http://skipforward.opendfki.de/wiki/DBTropes`

across different media. An experiment with articles from six UK and US news sites has shown that such deviations can be found for different topics, ranging from political parties to issues such as drug legislation and gay marriage [55].

Schulz et al. use the extension for developing an approach for semantic abstraction for generalization of tweets classification [271, 272]. Shidik et al. [278] make use of the extension for developing a machine learning approach for predicting forest fires using LOD as background knowledge. Lausch et al. [157] list the extension as a tool for performing data analysis in environmental research. Schaible et al. [266] use the extension for the task of movie classification using background knowledge from LOD.

[161] uses the tool to develop hybrid approach for automatic taxonomy learning, which combines a data-driven and a knowledge-based component. The approach uses the tool to extract structured semantic information from the Linked Open Data cloud (DBpedia) and WordNet.

The Semantic Web graph embeddings, RDF2Vec (described in Chapter 8, have been used in several other applications.

In [5], we use the graph embedding approach to exploit the vector representations of frames using the FrameNet [203] graph and the subsumption hierarchy of roles as represented in Framester [96], for the task of reconciling knowledge graphs extracted from text. The results show that using the graph embeddings significantly outperforms the baseline approaches on the given task.

The authors of [144], use RDF2Vec to develop a supervised algorithm for deriving type embeddings in the same latent space as a given set of entity embeddings. Given a set of pre-generated entity embeddings, and a sparse collection of type assertion triples (a subset of the ABox), the approach can generate embeddings for a set of types.

[103] presents an approach that can summarize facts about a collection of entities by analyzing their relatedness in preference to summarizing each entity in isolation. Specifically, the approach generates informative entity summaries by selecting: (i) inter-entity facts that are similar and (ii) intra-entity facts that are important and diverse, which are calculated using the RDF2vec approach.

The authors of [109], address the task of learning a model capable of classifying images into classes for which no sample is available as training data, by leverage the interlinking of knowledge bases published as Linked Open Data to provide different semantic feature representations of visual classes in a large-scale setting. To do so, the authors use the RDF2vec graph embeddings.

[193] presents an approach for building entity-centric event collections from large archives. The approach first identifies relevant concepts and entities from a knowledge base, and then detects their mentions in the documents, which are interpreted as indicators for relevance. For ranking the entities, the authors use RDF2Vec embeddings. The approach is compared to a set of baselines, and evaluated on three datasets, showing that RDF2vec outperforms all the baseline methods.

## 14.4 Open Issues and Limitations

As outlined in the individual chapters, there are several open issues and limitations of the introduced approaches, which bring opportunities for future work. Here we discuss general open issues of the thesis.

In this thesis we only focused on a limited set of data mining datasets and tasks. More precisely, most of the approaches are evaluated on standard data mining tasks. In future work, we would consider extending the evaluation to other tasks, such as link prediction, type prediction, graph completion and error detection in knowledge graphs, among the others. Furthermore, the benchmark of datasets used for evaluation can be extended to achieve more thorough evaluation of the approaches.

Another notable limitation of the thesis is that we are not able to directly measure the quality of the generated Semantic Web knowledge graph embeddings, i.e., the quality of the embeddings is measured through specific data mining tasks, extrinsic evaluation, which shows that there is no "one size fit all solution". In future work, we would consider developing an intrinsic evaluation benchmark, which would reduce the time for evaluating the approaches, and would make the evaluation process more transparent and easier to compare to other approaches.

Furthermore, while we set the basics for possible integration of multiple Semantic Web knowledge graphs to be used in data mining, we didn't evaluate to which extent this can be done with the graph embeddings approaches. To do so, we consider two approaches: (i) embed each graph separately, and then align the entities and properties, or (ii) perform the alignment and the embedding of the graphs in the same step.

## 14.5 Future Work

In this section we discuss possible future directions of work for exploiting Semantic Web knowledge graphs in data mining.

With the recent developments, both in research and industry, most of the data is represented in the form of knowledge graphs. As we already showed in this thesis, there are several initiatives in the research field that try to convert the whole human knowledge in large Semantic Web knowledge graphs, like DBpedia, Wikidata, and YAGO. Also, the IT giants represent their data in the form of knowledge graphs, e.g., Google Knowledge Grpah,[4] Yahoo Knowledge Graph[5] and IBM knowledge graph (used by IBM Watson)[6], among the others. This leads to a rapid development of approaches for managing such data. i.e., generating, consuming and mining.

[4] https://www.google.com/intl/bn/insidesearch/features/search/knowledge.html

[5] http://semtechbizsj2014.semanticweb.com/sessionPop.cfm?confid=82&proposalid=6452

[6] https://www.ibm.com/watson/

One of the research directions is converting the vast amount of unstructured data published on the Web in various formats, e.g., text, lists, tables, etc, into graph data. This data contains useful knowledge that aligned with existing Semantic Web knowledge graphs can be of a high value. Furthermore, maintaining high quality knowledge graphs is not trivial. To address this issues, data mining solutions can be used.

While, in this thesis we described a set of approaches for converting graph data in propositional form that can be directly used in existing data mining tools and algorithms, in the future work we believe that more effort will be put in developing data mining algorithms that will be applicable directly on graph data. Such algorithms would not follow the general KDD pipeline, i.e., the *Transformation* step will be omitted.

When developing such approaches, a lot of effort will go into optimizing the data mining algorithms, due to the large size of the knowledge graphs. Even today, the knowledge graphs can grow up to several million of nodes and billions of edges, which is pushing the currently available processing power to its limits. In many cases multiple graphs need to be merged together, which exponentially increases the computational complexity. This brings the attention to the need for developing efficient approaches for merging large-scale knowledge graphs.

To summarize, we believe that contributions like this thesis will lead to a rapid advancement in the technology for graph-based representation and consumption of data, which will integrate data mining as a non-separable part of the maintenance and consumption of the data.

# Bibliography


[1] Fabian Abel, Ilknur Celik, Geert-Jan Houben, and Patrick Siehndel. Leveraging the semantics of tweets for adaptive faceted search on twitter. In *The Semantic Web–ISWC 2011*, pages 1–17. Springer, 2011.

[2] Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. Semantic enrichment of twitter posts for user profile construction on the social web. In *The Semanic Web: Research and Applications*, pages 375–389. Springer, 2011.

[3] Nitish Aggarwal and Paul Buitelaar. Wikipedia-based distributional semantics for entity relatedness. In *2014 AAAI Fall Symposium Series*, 2014.

[4] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[5] Mehwish Alam, Diego Reforgiato Recupero, Misael Mongiovi, Aldo Gangemi, and Petar Ristoski. Event-based knowledge reconciliation using frame embeddings and frame similarity. *Knowledge-Based Systems*, 2017.

[6] Cláudia Antunes. Onto4ar: a framework for mining association rules. In *Workshop on Constraint-Based Mining and Learning (CMILE-ECML/PKDD 2007)*, pages 37–48, 2007.

[7] Claudia Antunes. An ontology-based framework for mining patterns in the presence of background knowledge. In *1st International Conference on Advanced Intelligence*, pages 163–168, 2008.

[8] Ghislain Auguste Atemezing and Raphaël Troncy. Towards a linked-data based visualization wizard. In *Workshop on Consuming Linked Data*, 2014.

[9] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.

[10] Andrea Bellandi, Barbara Furletti, Valerio Grossi, and Andrea Romei. Ontology-driven association rule extraction: A case study. *Contexts and Ontologies Representation and Reasoning*, page 10, 2007.

[11] Alejandro Bellogín, Iván Cantador, and Pablo Castells. A comparative study of heterogeneous item recommendations in social systems. *Inf. Sci.*, 221:142–169, February 2013.

[12] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[13] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[14] Abraham Bernstein, Foster Provost, and Shawndra Hill. Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):503–518, 2005.

[15] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. Methods for exploring and mining tables on wikipedia. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, IDEA '13, pages 18–26, New York, NY, USA, 2013. ACM.

[16] Veli Bicer, Thanh Tran, and Anna Gossen. Relational kernel machines for learning from graph-structured rdf data. In *The Semantic Web: Research and Applications*, pages 47–62. Springer, 2011.

[17] Ana Cristina Bicharra Garcia, Inhauma Ferraz, and Adriana s. Vivacqua. From data to knowledge mining. *Artif. Intell. Eng. Des. Anal. Manuf.*, 23(4):427–441, November 2009.

[18] Chris Biemann. Ontology learning from text: A survey of methods. *LDV Forum*, 20(2):75–93, 2005.

[19] Christian Bizer. D2rq - treating non-rdf databases as virtual rdf graphs. In *In Proceedings of the 3rd International Semantic Web Conference*, 2004.

[20] Christian Bizer, Richard Cyganiak, and Tobias Gauß. The rdf book mashup: from web apis to a web of data. In *Scripting for the Semantic Web*, 2007.

[21] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[22] Jens Bleiholder and Felix Naumann. Data fusion. *ACM*, 41(1), 2008.

[23] Stephan Bloehdorn and York Sure. Kernel methods for mining instance data in ontologies. In *Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, pages 58–71, Berlin, Heidelberg, 2007. Springer-Verlag.

[24] Peter Bloem, Adianto Wibisono, and Gerben de Vries. Simplifying rdf data for graph-based machine learning. In *11th ESWC 2014 (ESWC2014)*, May 2014.

[25] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *ARTIFICIAL INTELLIGENCE*, 97:245–271, 1997.

[26] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA, 2008. ACM.

[27] Kalina Bontcheva and Dominic Rout. Making sense of social media streams through semantics: a survey. *Semantic Web*, 1:1–31, 2012.

[28] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[29] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107 – 117, 1998.

[30] Stefan Brüggemann and Hans-Jürgen Appelrath. Context-aware replacement operations for data cleaning. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 1700–1704, New York, NY, USA, 2011. ACM.

[31] Stefan Brüggemann and Fabian Grüning. Using domain knowledge provided by ontologies for improving data quality management. In *Proceedings of I-Know*, pages 251–258, 2008.

[32] Josep Maria Brunetti, Sören Auer, and Roberto García. The linked data visualization model. In *International Semantic Web Conference (Posters & Demos)*, 2012.

[33] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[34] Margit Bussmann and Andreas Kleine. The analysis of european research networks: Cooperation in the seventh framework program. In *ECPR General Conference*, 2011.

[35] Peter Cabena, Pablo Hadjinian, Rolf Stadler, Jaap Verhees, and Alessandro Zanasi. *Discovering data mining: from concept to implementation*. Prentice-Hall, Inc., 1998.

[36] Michael J. Cafarella, Alon Halevy, and Nodira Khoussainova. Data Integration for the Relational Web. *Proc. VLDB Endow.*, 2(1):1090–1101, 2009.

[37] Iván Cantador, Ignacio Fernández-Tobıas, Shlomo Berkovsky, and Paolo Cremonesi. Cross-domain recommender systems. 2015.

[38] Hana Cešpivová, Jan Rauch, Vojtech Svatek, Martin Kejkula, and Marie Tomeckova. Roles of medical ontology in association mining crisp-dm cycle. In *ECML/PKDD04 Workshop on Knowledge Discovery and Ontologies (KDO'04), Pisa*, volume 220. Citeseer, 2004.

[39] Jeffrey Chan, Conor Hayes, and Elizabeth Daly. Decomposing discussion forums using common user roles. 2010.

[40] Mark Chatfield and Adrian Mander. The skillings–mack test (friedman test when there are missing data). *The Stata Journal*, 9(2):299, 2009.

[41] Jilin Chen, Rowan Nairn, Les Nelson, Michael Bernstein, and Ed Chi. Short and tweet: experiments on recommending content from information streams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1185–1194. ACM, 2010.

[42] Gong Cheng, Thanh Tran, and Yuzhong Qu. Relin: relatedness and informativeness-based centrality for entity summarization. In *The Semantic Web–ISWC*, pages 114–129. 2011.

[43] Weiwei Cheng, Gjergji Kasneci, Thore Graepel, David Stern, and Ralf Herbrich. Automated feature generation from structured knowledge. In *20th ACM Conference on Information and Knowledge Management (CIKM 2011)*, 2011.

[44] Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, and Heiko Paulheim. Biased graph walks for rdf graph embeddings. In *Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics*, page 21. ACM, 2017.

[45] Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, and Heiko Paulheim. Global rdf vector space embeddings. 2017.

[46] Allan M Collins and Elizabeth F Loftus. A spreading-activation theory of semantic processing. *Psychological review*, 82(6):407, 1975.

[47] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[48] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[49] Aba-Sah Dadzie and Matthew Rowe. Approaches to visualising linked data: A survey. *Semantic Web*, 2(2):89–124, 2011.

[50] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N Mendes. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 121–124. ACM, 2013.

[51] Mathieu d'Aquin and Nicolas Jay. Interpreting data mining results with linked data for learning analytics: Motivation, case study and directions. In *Proceedings of the Third International Conference on Learning Analytics and Knowledge*, LAK '13, pages 155–164, New York, NY, USA, 2013. ACM.

[52] Manoranjan Dash and Huan Liu. Feature selection for classification. *Intelligent data analysis*, 1(3):131–156, 1997.

[53] Victor de Boer, Jan Wielemaker, Judith van Gent, Michiel Hildebrand, Antoine Isaac, Jacco van Ossenbruggen, and Guus Schreiber. Supporting linked data production for cultural heritage institutes: The amsterdam museum case study. In *The Semantic Web: Research and Applications*, pages 733–747. Springer, 2012.

[54] Jean C de Borda. *Mémoire sur les élections au scrutin*. Histoire de l'Academie Royale des Sciences, 1781.

[55] Orphee De Clercq, Sven Hertling, Veronique Hoste, Simone Paolo Ponzetto, and Heiko Paulheim. Identifying disputed topics in the news. In *Linked Data for Knowledge Discovery (LD4KD)*, pages 37–48. CEUR, 2014.

[56] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Narducci Fedelucio, and Giovanni Semeraro. Semantics-aware content-based recommender systems. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 119–159. Springer, 2nd edition, 2015.

[57] Gerben Klaas Dirk de Vries. A fast approximation of the weisfeiler-lehman graph kernel for rdf data. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Zelezný, editors, *ECML/PKDD (1)*, volume 8188 of *Lecture Notes in Computer Science*, pages 606–621. Springer, 2013.

[58] Gerben Klaas Dirk de Vries and Steven de Rooij. A fast and simple graph kernel for rdf. In *Proceedings of the Second International Workshop on Knowledge Discovery and Data Mining Meets Linked Open Data*, 2013.

[59] Gerben Klaas Dirk de Vries and Steven de Rooij. Substructure counting graph kernels for machine learning from rdf data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2015.

[60] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[61] Zhi-Hong Deng, Shi-Wei Tang, Dong-Qing Yang, Ming Zhang Li-Yu Li, and Kun-Qing Xie. A comparative study on feature weight in text categorization. In *Advanced Web Technologies and Applications*, pages 588–597. Springer, 2004.

[62] T. Di Noia, V. C. Ostuni, P. Tomeo, and E. Di Sciascio. Sprank: Semantic path-based ranking for top-n recommendations using linked open data. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2016.

[63] Tommaso Di Noia, Iván Cantador, and Vito Claudio Ostuni. Linked open data-enabled recommender systems: Eswc 2014 challenge on book recommendation. In *Semantic Web Evaluation Challenge*, pages 129–143. Springer, 2014.

[64] Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, and Davide Romito. Exploiting the web of data in model-based recommender systems. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 253–256, New York, NY, USA, 2012. ACM.

[65] Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, Davide Romito, and Markus Zanker. Linked open data to support content-based recommender systems. In *Proceedings of the 8th International Conference on Semantic Systems*, I-SEMANTICS '12, pages 1–8, New York, NY, USA, 2012. ACM.

[66] Tommaso Di Noia and Vito Claudio Ostuni. Recommender systems and linked open data. In *Reasoning Web. Web Logic Rules*, pages 88–113. Springer, 2015.

[67] Tommaso Di Noia, Vito Claudio Ostuni, Jessica Rosati, Paolo Tomeo, Eugenio Di Sciascio, Roberto Mirizzi, and Claudio Bartolini. Building a relatedness graph from linked open data: A case study in the IT domain. *Expert Syst. Appl.*, 44:354–366, 2016.

[68] Claudia Diamantini and Domenico Potena. Semantic annotation and services for kdd tools sharing and reuse. In *ICDM Workshops*, pages 761–770, 2008.

[69] Claudia Diamantini, Domenico Potena, and Emanuele Storti. Kddonto: An ontology for discovery and composition of kdd algorithms. *Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery (SoKD'09)*, pages 13–24, 2009.

[70] Claudia Diamantini, Domenico Potena, and Emanuele Storti. Supporting users in kdd processes design: a semantic similarity matching approach. In *Planning to Learn Workshop (PlanLearn'10) at ECAI*, pages 27–34, 2010.

[71] Li Ding, Dominic DiFranzo, Alvaro Graves, James Michaelis, Xian Li, Deborah L. McGuinness, and James A. Hendler. Twc data-gov corpus: incrementally generating linked government data from data.gov. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *WWW*, pages 1383–1386. ACM, 2010.

[72] Marcos Aurélio Domingues and Solange Oliveira Rezende. Using taxonomies to facilitate the analysis of the association rules. *arXiv preprint arXiv:1112.1734*, 2011.

[73] Dejing Dou, Hao Wang, and Haishan Liu. Semantic data mining: A survey of ontology-based approaches. In *Semantic Computing (ICSC), 2015 IEEE International Conference on*, pages 244–251. IEEE, 2015.

[74] Sašo Džeroski. *Towards a General Framework for Data Mining*. KDID'06. Springer-Verlag, Berlin, Heidelberg, 2007.

[75] Sašo Džeroski. *Relational Data Mining*, pages 887–911. Springer US, Boston, MA, 2010.

[76] Jacob Eisenstein, Duen Horng Chau, Aniket Kittur, Eric P Xing, et al. Topicviz: Semantic navigation of document collections. *arXiv preprint arXiv:1110.6200*, 2011.

[77] Andrew F Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong. Systematic construction of anomaly detection benchmarks from real data. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, pages 16–21. ACM, 2013.

[78] Lauri Eronen and Hannu Toivonen. Biomine: predicting links between biological entities using network models of heterogeneous databases. *BMC bioinformatics*, 13(1):119, 2012.

[79] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[80] Nicola Fanizzi and Claudia d'Amato. A declarative kernel for alc concept descriptions. In *Foundations of Intelligent Systems*, pages 322–331. Springer, 2006.

[81] Nicola Fanizzi and Floriana Esposito. Statistical learning for inductive query answering on owl ontologies. In *In Proceedings of the 7th International Semantic Web Conference (ISWC*, 2008.

[82] S. Faralli, A. Panchenko, C. Biemann, and S. P. Ponzetto. Linked disambiguated distributional semantic networks. In *ISWC*, pages 56–64, 2016.

[83] S. Faralli, A. Panchenko, C. Biemann, and S. P. Ponzetto. The contrastmedium algorithm: Taxonomy induction from noisy knowledge graphs with just a few links. In *EACL (to appear)*, 2017. https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2017-farallietal-eacl-contrastmedium.pdf.

[84] Ines Färber, Stephan Günnemann, Hans-Peter Kriegel, Peer Kröger, Emmanuel Müller, Erich Schubert, Thomas Seidl, and Arthur Zimek. On using class-labels in evaluation of clusterings. In *MultiClust: Workshop on Discovering, Summarizing and Using Multiple Clusterings*, 2010.

[85] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. Advances in knowledge disc overy and data mining. pages 1–34, Menlo Park, CA, USA, 1996. American Association for Artificial Intelligence.

[86] Ignacio Fernández-Tobías, Iván Cantador, Marius Kaminskas, and Francesco Ricci. A generic semantic-based framework for cross-domain recommendation. In *Proceedings of the 2Nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '11, pages 25–32, New York, NY, USA, 2011. ACM.

[87] Tim Finin and Zareen Syed. Creating and exploiting a web of semantic data. In *ICAART (1)*, pages 7–18, 2010.

[88] Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. Learning semantic hierarchies via word embeddings. In *ACL (1)*, pages 1199–1209, 2014.

[89] Christian Fürber and Martin Hepp. Using semantic web resources for data quality management. In *Proceedings of the 17th International Conference on Knowledge Engineering and Management by the Masses*, EKAW'10, pages 211–225, Berlin, Heidelberg, 2010. Springer-Verlag.

[90] Christian Fürber and Martin Hepp. Using sparql and spin for data quality management on the semantic web. In *Business Information Systems*, pages 35–46. Springer, 2010.

[91] Christian Fürber and Martin Hepp. Swiqa-a semantic web information quality assessment framework. In *ECIS*, volume 15, page 19, 2011.

[92] Christian Fürber and Martin Hepp. Using semantic web technologies for data quality management. In *Handbook of data quality*, pages 141–161, 2013.

[93] Johannes Fürnkranz. Separate-and-Conquer Rule Learning. *Artificial Intelligence Review*, 13(1):3–54, 1999.

[94] Alexander Gabriel, Heiko Paulheim, and Frederik Janssen. Learning semantically coherent rules. In *Interactions between Data Mining and Natural Language Processing*, pages 49–63, 2014.

[95] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJcAI*, volume 7, pages 1606–1611, 2007.

[96] Aldo Gangemi, Mehwish Alam, Luigi Asprino, Valentina Presutti, and Diego Reforgiato Recupero. Framester: A wide coverage linguistic linked data hub. In *Knowledge Engineering and Knowledge Management - 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings*, pages 239–254, 2016.

[97] Ana Cristina Bicharra Garcia and Adriana S Vivacqua. Does ontology help make sense of a complex world or does it create a biased interpretation? In *Proc. Sensemaking Workshop in CHI*, volume 8, 2008.

[98] Jyoti Gautam and Ela Kumar. An integrated and improved approach to terms weighting in text classification. *IJCSI International Journal of Computer Science Issues*, 10(1), 2013.

[99] Robert L. Glass and Iris Vessey. Contemporary application-domain taxonomies. *IEEE Software*, 12(4):63–76, 1995.

[100] Markus Goldstein. Anomaly detection. In *RapidMiner – Data Mining Use Cases and Business Analytics Applications*. 2014.

[101] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864. ACM, 2016.

[102] Thomas R Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International journal of human-computer studies*, 43(5):907–928, 1995.

[103] Kalpa Gunaratna, Amir Hossein Yazdavar, Krishnaprasad Thirunarayan, Amit Sheth, and Gong Cheng. Relatedness-based multi-entity summarization. 2017.

[104] Niharika Gupta, Sanjay Podder, KM Annervaz, and Shubhashis Sengupta. Domain ontology induction using word embeddings. In *Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on*, pages 115–119. IEEE, 2016.

[105] Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(Feb):307–361, 2012.

[106] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[107] Lushan Han, Tim Finin, Cynthia Parr, Joel Sachs, and Anupam Joshi. Rdf123: From spreadsheets to rdf. In *Proceedings of the 7th International Conference on The Semantic Web*, ISWC '08, pages 451–466, Berlin, Heidelberg, 2008. Springer-Verlag.

[108] David Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*. MIT Press, 2001.

[109] Tristan Hascoet, Yasuo Ariki, and Tetsuya Takiguchi. Semantic web and zero-shot learning of large scale visual classes.

[110] Samer Hassan and Rada Mihalcea. Semantic relatedness using salient semantic analysis. In *AAAI*, 2011.

[111] Oktie Hassanzadeh, Soheil Hassas Yeganeh, and Renée J. Miller. Linking semistructured data on the web. In *WebDB*, 2011.

[112] Tuukka Hastrup, Richard Cyganiak, and Uldis Bojars. Browsing linked data with fenfire. 2008.

[113] M. A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proc. of the COLING-Volume 2*, pages 539–545, 1992.

[114] Dominik Heckmann, Tim Schwartz, Boris Brandherm, Michael Schmitz, and Margeritta von Wilamowitz-Moellendorff. Gumo–the general user model ontology. In *User modeling 2005*, pages 428–432. Springer, 2005.

[115] Benjamin Heitmann, Maciej Dabrowski, Alexandre Passant, Conor Hayes, and Keith Griffin. Personalisation of social web services in the enterprise using spreading activation for multi-source, cross-domain recommendations. In *AAAI Spring Symposium: Intelligent Web Services Meet Social Computing*, 2012.

[116] Benjamin Heitmann and Conor Hayes. Using linked data to build open, collaborative recommender systems. In *In: AAAI Spring Symposium: Linked Data Meets Artificial Intelligence'. (2010*, 2010.

[117] Benjamin Heitmann and Conor Hayes. Semstim at the lod-recsys 2014 challenge. In Valentina Presutti, Milan Stankovic, Erik Cambria, Iván Cantador, Angelo Di Iorio, Tommaso Di Noia, Christoph Lange, Diego Reforgiato Recupero, and Anna Tordai, editors, *Semantic Web Evaluation Challenge*, volume 475 of *Communications in Computer and Information Science*, pages 170–175. Springer International Publishing, 2014.

[118] Daniel Hienert, Dennis Wegener, and Heiko Paulheim. Automatic classification and relationship extraction for multi-lingual and multi-granular events from wikipedia. *Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2012)*, 902:1–10, 2012.

[119] Melanie Hilario, Alexandros Kalousis, Phong Nguyen, and Adam Woznica. A data mining ontology for algorithm selection and meta-mining. In *Proceedings of the ECML/PKDD09 Workshop on 3rd generation Data Mining (SoKD-09)*, pages 76–87, 2009.

[120] Melanie Hilario, Phong Nguyen, Huyen Do, Adam Woznica, and Alexandros Kalousis. Ontology-based meta-mining of knowledge discovery workflows. In *Meta-Learning in Computational Intelligence*, pages 273–315. Springer, 2011.

[121] Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. Kore: keyphrase overlap relatedness for entity disambiguation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 545–554. ACM, 2012.

[122] Markus Hofmann and Ralf Klinkenberg. *RapidMiner: Data Mining Use Cases and Business Analytics Applications*. CRC Press, 2013.

[123] Geoffrey Holmes, Mark Hall, and Eibe Prank. *Generating rule sets from model trees*. Springer, 1999.

[124] Ian Horrocks, Peter F Patel-Schneider, and Frank Van Harmelen. From shiq and rdf to owl: The making of a web ontology language. *Web semantics: science, services and agents on the World Wide Web*, 1(1):7–26, 2003.

[125] Lan Huang, David Milne, Eibe Frank, and Ian H Witten. Learning a concept-based document similarity measure. *Journal of the American Society for Information Science and Technology*, 63(8):1593–1608, 2012.

[126] Yi Huang, Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A scalable kernel approach to learning in semantic graphs with applications to linked data. In *1st Workshop on Mining the Future Internet*, 2010.

[127] Yi Huang, Volker Tresp, Markus Bundschus, Achim Rettinger, and Hans-Peter Kriegel. Multivariate prediction for learning on the semantic web.

In Paolo Frasconi and FrancescaA. Lisi, editors, *Inductive Logic Programming*, volume 6489 of *Lecture Notes in Computer Science*, pages 92–104. Springer Berlin Heidelberg, 2011.

[128] Yi Huang, Volker Tresp, Maximilian Nickel, Achim Rettinger, and Hans-Peter Kriegel. A scalable approach for statistical learning in semantic graphs. *Semantic Web*, 5(1):5–22, 2014.

[129] Zhaohui Huang, Huajun Chen, Tong Yu, Hao Sheng, Zhaobo Luo, and Yuxin Mao. Semantic text mining with linked data. In *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*, pages 338–343, Aug 2009.

[130] Ioana Hulpuş, Narumol Prangnawarat, and Conor Hayes. Path-based semantic relatedness on linked data and its use to word and entity disambiguation. In *International Semantic Web Conference*, pages 442–457. Springer, 2015.

[131] David Huynh, Stefano Mazzocchi, and David Karger. Piggy bank: Experience the semantic web inside your web browser. In *The Semantic Web–ISWC 2005*, pages 413–430. Springer, 2005.

[132] Nicolas Jay and Mathieu d'Aquin. Linked data and online classifications to organise mined patterns in patient data. In *AMIA Annual Symposium Proceedings*, volume 2013, page 681. American Medical Informatics Association, 2013.

[133] Yoonjae Jeong and Sung-Hyon Myaeng. Feature selection using a semantic hierarchy for event recognition and type classification. In *International Joint Conference on Natural Language Processing*, 2013.

[134] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *ICML'94*, pages 121–129, 1994.

[135] Milos Jovanovik, Aleksandra Bogojeska, Dimitar Trajanov, and Ljupco Kocarev. Inferring cuisine-drug interactions using the linked data approach. *Scientific reports*, 5, 2015.

[136] Tobias Käfer and Andreas Harth. Billion Triples Challenge data set. Downloaded from http://km.aifb.kit.edu/projects/btc-2014/, 2014.

[137] Marius Kaminskas, Ignacio ernández Tobıas, Francesco Ricci, and Iván Cantador. Knowledge-based identification of music suited for places of interest. *Information Technology & Tourism*, 14(1):73–95, 2014.

[138] Marius Kaminskas, Ignacio Fernández-Tobías, Iván Cantador, and Francesco Ricci. Ontology-based identification of music for places. In Lorenzo Cantoni and Zheng (Phil) Xiang, editors, *Information and Communication Technologies in Tourism 2013*, pages 436–447. Springer Berlin Heidelberg, 2013.

[139] Benedikt Kämpgen and Andreas Harth. Olap4ld - a framework for building analysis applications over governmental statistics. In *ESWC 2014 Posters & Demo session*. ESWC, Springer, Mai 2014.

[140] Venkata Narasimha Pavan Kappara, Ryutaro Ichise, and O.P. Vyas. Liddm: A data mining system for linked data. In *Workshop on Linked Data on the Web (LDOW2011)*, 2011.

[141] Tomi Kauppinen, Giovana Mira de Espindola, and Benedikt Gräler. Sharing and analyzing remote sensing observation data for linked science. In *Poster proceedings of the Extended Semantic Web Conference*. Citeseer, 2012.

[142] Tomi Kauppinen, Giovana Mira de Espindola, Jim Jones, Alber Sánchez, Benedikt Gräler, and Thomas Bartoschek. Linked brazilian amazon rainforest data. *Semantic Web*, 5(2):151–155, 2014.

[143] Zoubida Kedad and Elisabeth Métais. Ontology-based data cleaning. In *Natural Language Processing and Information Systems*, pages 137–149. Springer, 2002.

[144] Mayank Kejriwal and Pedro Szekely. Scalable generation of type embeddings using the abox. *Open Journal of Semantic Web (OJSW)*, 4(1):20–34, 2017.

[145] Mansoor Ahmed Khan, Gunnar Aastrand Grimnes, and Andreas Dengel. Two pre-processing operators for improved learning from semanticweb data. In *First RapidMiner Community Meeting And Conference (RCOMM 2010)*, volume 20, 2010.

[146] J Kietz, Floarea Serban, Abraham Bernstein, and Simon Fischer. Towards cooperative planning of data mining workflows. In *Proceedings of the Third Generation Data Mining Workshop at the 2009 European Conference on Machine Learning (ECML 2009)*, pages 1–12, 2009.

[147] Jörg-Uwe Kietz, Floarea Serban, and Abraham Bernstein. eproplan: A tool to model automatic generation of data mining workflows. In *Proceedings of the 3rd Planning to Learn Workshop (WS9) at ECAI*, volume 2010, 2010.

[148] Jörg-Uwe Kietz, Floarea Serban, Abraham Bernstein, and Simon Fischer. Data mining workflow templates for intelligent discovery assistance and auto-experimentation. *Third-Generation Data Mining: Towards Service-oriented Knowledge Discovery (SoKD-10)*, pages 1–12, 2010.

[149] Jörg-Uwe Kietz, Floarea Serban, Simon Fischer, and Abraham Bernstein. "semantics inside!" but let's not tell the data miners: Intelligent support for data mining. In *The Semantic Web: Trends and Challenges*, pages 706–720. Springer, 2014.

[150] Jakub Klímek, Jirí Helmich, and M Neasky. Application of the linked data visualization model on real world data from the czech lod cloud. In *6th International Workshop on the Linked Data on the Web (LDOW'14)*, 2014.

[151] Willi Klösgen. Knowledge discovery in databases and data mining. In *Foundations of Intelligent Systems*, volume 1079 of *Lecture Notes in Computer Science*, pages 623–632. Springer Berlin Heidelberg, 1996.

[152] Xiangnan Kong and Philip S. Yu. An ensemble-based approach to fast classification of multi-label data streams. In *CollaborateCom*, pages 95–104, 2011.

[153] Zornitsa Kozareva and Eduard Hovy. A semi-supervised method to learn and construct taxonomies using the web. In *EMNLP*, pages 1110–1118, 2010.

[154] Stefan Kramer, Nada Lavrač, and Peter Flach. Propositionalization approaches to relational data mining. In Sašo Džeroski and Nada Lavrač, editors, *Relational Data Mining*, pages 262–291. Springer Berlin Heidelberg, 2001.

[155] Man Lan, Chew-Lim Tan, Hwee-Boon Low, and Sam-Yuan Sung. A comprehensive comparative study on term weighting schemes for text categorization with support vector machines. In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, WWW '05, pages 1032–1033, New York, NY, USA, 2005. ACM.

[156] Andreas Langegger and Wolfram Wöß. Xlwrap – querying and integrating arbitrary spreadsheets with sparql. In Abraham Bernstein, DavidR. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, editors, *The Semantic Web - ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 359–374. Springer Berlin Heidelberg, 2009.

[157] Angela Lausch, Andreas Schmidt, and Lutz Tischendorf. Data mining and linked open data–new perspectives for data analysis in environmental research. *Ecological Modelling*, 295:5–17, 2015.

[158] Nada Lavrač and Petra Kralj Novak. Relational and semantic data mining for biomedical research, 2012.

[159] Nada Lavrač, Anže Vavpetič, Larisa Soldatova, Igor Trajkovski, and Petra Kralj Novak. Using ontologies in semantic data mining with segs and g-segs. In *Proceedings of the 14th International Conference on Discovery Science*, DS'11, pages 165–178, Berlin, Heidelberg, 2011. Springer-Verlag.

[160] M Lee, Brandon Pincombe, and Matthew Welsh. An empirical evaluation of models of text document similarity. Cognitive Science Society, 2005.

[161] Els Lefever. A hybrid approach to domain-independent taxonomy learning. *Applied Ontology*, 11(3):255–278, 2016.

[162] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 2013.

[163] Oliver Lehmberg, Dominique Ritze, Petar Ristoski, Robert Meusel, Heiko Paulheim, and Christian Bizer. The mannheim search join engine. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2015.

[164] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

[165] Siaw-Teng Liaw, A. Rahimi, P. Ray, Jane Taggart, S. Dennis, Simon de Lusignan, B. Jalaludin, A. E. T. Yeo, and A. Talaei-Khoei. Towards an ontology for data quality in integrated chronic disease management: A realist review of the literature. *I. J. Medical Informatics*, 82(1):10–24, 2013.

[166] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proc. VLDB Endow.*, 3(1-2):1338–1347, September 2010.

[167] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187, 2015.

[168] Haishan Liu. Towards semantic data mining. In *In Proc. of the 9th International Semantic Web Conference (ISWC2010*, 2010.

[169] Uta Lösch, Stephan Bloehdorn, and Achim Rettinger. Graph kernels for rdf data. In *The Semantic Web: Research and Applications*, pages 134–148. Springer, 2012.

[170] Uta Lösch, Stephan Bloehdorn, and Achim Rettinger. Graph kernels for RDF data. In *ESWC*, pages 134–148, 2012.

[171] Sisi Lu, Ye Ye, Rich Tsui, Howard Su, Ruhsary Rexit, Sahawut Wesaratchakit, Xiaochu Liu, and Rebecca Hwa. Domain ontology-based feature reduction for high dimensional drug data and its application to 30-day heart failure readmission prediction. In *International Conference Conference on Collaborative Computing (Collaboratecom)*, pages 478–484, 2013.

[172] Tomasz Lukaszewski. Extending bayesian classifier with ontological attributes. 2010.

[173] Claudia Marinica and Fabrice Guillet. Knowledge-based interactive post-mining of association rules using ontologies. *Knowledge and Data Engineering, IEEE Transactions on*, 22(6):784–797, 2010.

[174] Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems (I-Semantics)*, 2011.

[175] Robert Meusel, Petar Petrovski, and Christian Bizer. The webdatacommons microdata, rdfa and microformat dataset series. In *The Semantic Web–ISWC*, pages 277–292. 2014.

[176] M. Mihelčić, N. Antulov-Fantulin, M. Bošnjak, and T. Šmuc. Extending rapidminer with recommender systems algorithms. In *RapidMiner Community Meeting and Conference (RCOMM 2012)*, 2012.

[177] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[178] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[179] Diego Milano, Monica Scannapieco, and Tiziana Catarci. Using ontologies for xml data cleaning. In *Proceedings of the 2005 OTM Confederated International Conference on On the Move to Meaningful Internet Systems*, OTM'05, pages 562–571, Berlin, Heidelberg, 2005. Springer-Verlag.

[180] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[181] Pasquale Minervini, Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. Scalable learning of entity and predicate embeddings for knowledge graph completion. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 162–167. IEEE, 2015.

[182] Luis Carlos Molina, Lluís Belanche, and Àngela Nebot. Feature selection algorithms: A survey and experimental evaluation. In *International Conference on Data Mining (ICDM)*, pages 306–313. IEEE, 2002.

[183] Laura Moss, David Corsar, and Ian Piper. A linked data approach to assessing medical data. In *Computer-Based Medical Systems (CBMS), 2012 25th International Symposium on*, pages 1–4. IEEE, 2012.

[184] Emir Muñoz, Aidan Hogan, and Alessandra Mileo. Using linked data to mine rdf from wikipedia's tables. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 533–542, New York, NY, USA, 2014. ACM.

[185] Varish Mulwad. DC Proposal: Graphical Models and Probabilistic Reasoning for Generating Linked Data from Tables. In *Proceedings of Tenth International Semantic Web Conference, Part II*, LCNS, page 317–324. LCNS 7032, Springer-Verlag, October 2011. Submitted at the Doctoral Consortium.

[186] Varish Mulwad, Tim Finin, and Anupam Joshi. Semantic Message Passing for Generating Linked Data from Tables. In *Proceedings of the 12th International Semantic Web Conference*. Springer, October 2013.

[187] Varish Mulwad, Tim Finin, and Anupam Joshi. Interpreting Medical Tables as Linked Data to Generate Meta-Analysis Reports. In *Proceedings of the 15th IEEE International Conference on Information Reuse and Integration*. IEEE Computer Society, August 2014.

[188] Varish Mulwad, Tim Finin, Zareen Syed, and Anupam Joshi. T2LD: interpreting and representing tables as linked data. In *Proceedings of the ISWC 2010 Posters & Demonstrations Track: Collected Abstracts, Shanghai, China, November 9, 2010*, 2010.

[189] Varish Mulwad, Tim Finin, Zareen Syed, and Anupam Joshi. Using linked data to interpret tables. In *In Proc. 1st Int. Workshop on Consuming Linked Data*, 2010.

[190] Emir Muñoz, Aidan Hogan, and Alessandra Mileo. Triplifying wikipedia's tables. In *LD4IE@ISWC'13*, pages –1–1, 2013.

[191] Belgin Mutlu, Patrick Hoefler, Gerwald Tschinkel, EE Veas, Vedran Sabol, Florian Stegmaier, and Michael Granitzer. Suggesting visualisations for published data. *Proceedings of IVAPP*, pages 267–275, 2014.

[192] Jindřich Mynarz and Vojtěch Svátek. Towards a benchmark for lod-enhanced knowledge discovery from structured data. In *Proceedings of the Second International Workshop on Knowledge Discovery and Data Mining Meets Linked Open Data*, pages 41–48. CEUR-WS, 2013.

[193] Federico Nanni, Simone Paolo Ponzetto, and Laura Dietz. Building entity-centric event collections. In *Digital Libraries (JCDL), 2017 ACM/IEEE Joint Conference on*, pages 1–10. IEEE, 2017.

[194] PB Nemenyi. Distribution-free multiple comparisons." vol. *PhD: Princeton University*, 1963.

[195] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs: From multi-relational link prediction to automated knowledge graph construction. *arXiv preprint arXiv:1503.00759*, 2015.

[196] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

[197] Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. Holographic embeddings of knowledge graphs. In *AAAI*, pages 1955–1961, 2016.

[198] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816, 2011.

[199] Hctor Oscar Nigro, Sandra Gonzalez Cisaro, and Daniel Hugo Xodo. *Data Mining With Ontologies: Implementations, Findings and Frameworks*. Information Science Reference - Imprint of: IGI Publishing, Hershey, PA, 2007.

[200] Andreas Nolle, German Nemirovski, A Sicilia, and J Pleguezuelos. An approach for accessing linked open data for data mining purposes. In *Proceedings of RapidMiner Community Meeting and Conference (RCOMM 2013), Porto*, 2013.

[201] Petra Kralj Novak, Anže Vavpetič, Igor Trajkovski, and Nada Lavrač. N.: Towards semantic data mining with g-segs. In *In: Proceedings of the 11th International Multiconference Information Society (IS*, 2009.

[202] Bernardo Pereira Nunes, Ricardo Kawase, Besnik Fetahu, Stefan Dietze, Marco A Casanova, and Diana Maynard. Interlinking documents based on semantic graphs. *Procedia Computer Science*, 22:231–240, 2013.

[203] Andrea Giovanni Nuzzolese, Aldo Gangemi, and Valentina Presutti. Gathering lexical linked data and knowledge patterns from FrameNet. In *Proceedings of the 6th International Conference on Knowledge Capture (K-CAP 2011), June 26-29, 2011, Banff, Alberta, Canada*, pages 41–48, 2011.

[204] Marilena Oita, Antoine Amarilli, and Pierre Senellart. Cross-fertilizing deep web analysis and ontology enrichment. In Marco Brambilla, Stefano Ceri, Tim Furche, and Georg Gottlob, editors, *VLDS*, pages 5–8. CEUR-WS.org.

[205] V. C. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In *ACM RecSys '13*, pages 85–92, 2013.

[206] Vito Claudio Ostuni, Tommaso Di Noia, Roberto Mirizzi, and Eugenio Di Sciascio. Top-n recommendations from implicit feedback leveraging linked open data. In *IIR*, pages 20–27, 2014.

[207] VitoClaudio Ostuni, Tommaso Di Noia, Roberto Mirizzi, and Eugenio Di Sciascio. A linked data recommender system using a neighborhood-based graph kernel. In Martin Hepp and Yigal Hoffner, editors, *E-Commerce and Web Technologies*, volume 188 of *Lecture Notes in Business Information Processing*, pages 89–100. Springer International Publishing, 2014.

[208] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[209] Ding Pan and Yan Pan. Using ontology repository to support data mining. In *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, volume 2, pages 5947–5951. IEEE, 2006.

[210] Ding Pan, Jun-Yi Shen, and Mu-Xin Zhou. Incorporating domain knowledge into data mining process: An ontology based framework. *Wuhan University Journal of Natural Sciences*, 11(1):165–169, 2006.

[211] Tan Pang-Ning, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson, 2006.

[212] Pance Panov, Saso Dzeroski, and Larisa N Soldatova. Ontodm: An ontology of data mining. In *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*, pages 752–760. IEEE, 2008.

[213] Panče Panov, Sašo Džeroski, and Larisa N Soldatova. Representing entities in the ontodm data mining ontology. In *Inductive Databases and Constraint-Based Data Mining*, pages 27–58. Springer, 2010.

[214] Panče Panov, Larisa Soldatova, and Sašo Džeroski. Ontodm-kdd: Ontology for representing the knowledge discovery process. In *Discovery Science*, pages 126–140. Springer, 2013.

[215] Panče Panov, Larisa Soldatova, and Sašo Džeroski. Ontology of core data mining entities. *Data Mining and Knowledge Discovery*, 28(5-6):1222–1265, 2014.

[216] Alexandre Passant. dbrec — music recommendations using dbpedia. In *The Semantic Web – ISWC 2010*, volume 6497 of *Lecture Notes in Computer Science*, pages 209–224. Springer Berlin Heidelberg, 2010.

[217] Alexandre Passant and Philippe Laublet. Meaning of a tag: A collaborative approach to bridge the gap between tagging and linked data. In *LDOW*, 2008.

[218] Christian Paul, Achim Rettinger, Aditya Mogadala, Craig A Knoblock, and Pedro Szekely. Efficient graph-based document similarity. In *International Semantic Web Conference*, pages 334–349. Springer, 2016.

[219] Heiko Paulheim. Generating possible interpretations for statistics from linked open data. In *9th Extended Semantic Web Conference (ESWC)*, 2012.

[220] Heiko Paulheim. Nobody wants to live in a cold city where no music has been recorded. In *The Semantic Web: ESWC 2012 Satellite Events*, pages 387–391. Springer, 2012.

[221] Heiko Paulheim. Exploiting linked open data as background knowledge in data mining. In *Workshop on Data Mining on Linked Open Data*, 2013.

[222] Heiko Paulheim. Identifying wrong links between datasets by multi-dimensional outlier detection. In *Workshop on Debugging Ontologies and Ontology Mappings (WoDOOM), 2014*, 2014.

[223] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017.

[224] Heiko Paulheim and Christian Bizer. Type inference on noisy rdf data. In *International Semantic Web Conference*, pages 510–525, 2013.

[225] Heiko Paulheim and Johannes Fümkranz. Unsupervised generation of data mining features from linked open data. In *Proceedings of the 2nd international conference on web intelligence, mining and semantics*, page 31. ACM, 2012.

[226] Heiko Paulheim and Sven Hertling. Disoverability of sparql endpoints in linked open data. In *International Semantic Web Conference, Posters and Demos*, 2013.

[227] Heiko Paulheim and Robert Meusel. A Decomposition of the Outlier Detection Problem into a Set of Supervised Learning Problems. *Machine Learning*, (2-3):509–531, 2015.

[228] Heiko Paulheim and Florian Probst. Ontology-enhanced user interfaces: A survey. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 6(2):36–59, 2010.

[229] Heiko Paulheim, Petar Ristoski, Evgeny Mitichkin, and Christian Bizer. Data mining with background knowledge from the web. *RapidMiner World*, 2014.

[230] Oscar Peña, Unai Aguilera, and Diego López-de Ipiña. Linked open data visualization revisited: A survey. *Semantic Web Journal*, 2014.

[231] Marco Pennacchiotti and Ana-Maria Popescu. A machine learning approach to twitter user classification. *ICWSM*, 11:281–288, 2011.

[232] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[233] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[234] Joseph Phillips and Bruce G Buchanan. Ontology-guided knowledge discovery in databases. In *Proceedings of the 1st international conference on Knowledge capture*, pages 123–130. ACM, 2001.

[235] Filipe Mota Pinto and Manuel Filipe Santos. Considering application domain ontologies for data mining. *WSEAS Transactions on Information Science and Applications*, 6(9):1478–1492, 2009.

[236] John C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING, 1998.

[237] Vid Podpecan, Nada Lavrac, Igor Mozetic, Petra Kralj Novak, Igor Trajkovski, Laura Langohr, Kimmo Kulovesi, Hannu Toivonen, Marko Petek, Helena Motaln, and Kristina Gruden. Segmine workflows for semantic microarray data analysis in orange4ws. *BMC Bioinformatics*, pages 416–416, 2011.

[238] Vid Podpečan, Monika Zemenova, and Nada Lavrač. Orange4ws environment for service-oriented data mining. *The Computer Journal*, page bxr077, 2011.

[239] Jedrzej Potoniec and Agnieszka Lawrynowicz. Rmonto: ontological extension to rapidminer. In *10th International Semantic Web Conference*, 2011.

[240] David Pérez-Rey, Alberto Anguita, and José Crespo. Ontodataclean: Ontology-based integration and preprocessing of distributed data. In Nicos Maglaveras, Ioanna Chouvarda, Vassilis Koutkias, and Rüdiger W. Brause, editors, *ISBMDA*, volume 4345 of *Lecture Notes in Computer Science*, pages 262–272. Springer, 2006.

[241] Lizhen Qu, Christof Müller, and Iryna Gurevych. Using tag semantic network for keyphrase extraction in blogs. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 1381–1382. ACM, 2008.

[242] Qudamah K Quboa and Mohamad Saraee. A state-of-the-art survey on semantic web mining. *Intelligent Information Management*, 5:10, 2013.

[243] J. Ross Quinlan. Combining instance-based and model-based learning. In *ICML*, page 236, 1993.

[244] Achim Rettinger, Uta Lösch, Volker Tresp, Claudia d'Amato, and Nicola Fanizzi. Mining the semantic web. *Data Mining and Knowledge Discovery*, 24(3):613–662, 2012.

[245] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer-Verlag New York, Inc., New York, NY, USA, 2nd edition, 2015.

[246] Petar Ristoski, Christian Bizer, and Heiko Paulheim. Mining the web of linked data with rapidminer. *Web Semantics: Science, Services and Agents on the World Wide Web*, 35:142–151, 2015.

[247] Petar Ristoski, Gerben Klaas Dirk de Vries, and Heiko Paulheim. A collection of benchmark datasets for systematic evaluations of machine learning on the semantic web. In *International Semantic Web Conference (To Appear)*. Springer, 2016.

[248] Petar Ristoski, Stefano Faralli, Simone Paolo Ponzetto, and Heiko Paulheim. Large-scale taxonomy induction using entity and word embeddings. In *Proceedings of the International Conference on Web Intelligence*, pages 81–87. ACM, 2017.

[249] Petar Ristoski, Eneldo Loza Mencía, and Heiko Paulheim. A hybrid multi-strategy recommender system using linked open data. In *Semantic Web Evaluation Challenge*, pages 150–156. Springer, 2014.

[250] Petar Ristoski and Heiko Paulheim. Visual analysis of statistical data on maps using linked open data. In *The 12th Extented Semantic Web Conference (ESWC2015)*.

[251] Petar Ristoski and Heiko Paulheim. Analyzing statistics with background knowledge from linked open data. In *Workshop on Semantic Statistics*, 2013.

[252] Petar Ristoski and Heiko Paulheim. A comparison of propositionalization strategies for creating features from linked open data. In *Linked Data for Knowledge Discovery*, 2014.

[253] Petar Ristoski and Heiko Paulheim. Feature selection in hierarchical feature spaces. In *Discovery Science*, 2014.

[254] Petar Ristoski and Heiko Paulheim. Rdf2vec: Rdf graph embeddings for data mining. In *International Semantic Web Conference (To Appear)*. Springer, 2016.

[255] Petar Ristoski and Heiko Paulheim. Semantic web in data mining and knowledge discovery: A comprehensive survey. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2016.

[256] Petar Ristoski, Heiko Paulheim, Vojtech Svátek, and Vaclav Zeman. The linked data mining challenge 2015. In *KNOW@LOD*, 2015.

[257] Petar Ristoski, Heiko Paulheim, Vojtech Svátek, and Vaclav Zeman. The linked data mining challenge 2016. In *KNOWLOD*, 2016.

[258] Petar Ristoski, Jessica Rosati, Tommaso Di Noia, Renato De Leone, and Heiko Paulheim. Rdf2vec: Rdf graph embeddings and their applications. *Semantic Web*, 2017.

[259] Petar Ristoski, Michael Schuhmacher, and Heiko Paulheim. Using graph metrics for linked open data enabled recommender systems. In *International Conference on Electronic Commerce and Web Technologies*, pages 30–41. Springer, 2015.

[260] Giuseppe Rizzo and Raphaël Troncy. Nerd: a framework for unifying named entity recognition and disambiguation extraction tools. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 73–76. Association for Computational Linguistics, 2012.

[261] Jessica Rosati, Petar Ristoski, Tommaso Di Noia, Renato De Leone, and Heiko Paulheim. Rdf graph embeddings for content-based recommender systems. In *Proceedings of the 3rd Workshop on New Trends in Content-based Recommender Systems (CBRecSys 2016)*, September 2016.

[262] Jeffrey D Sachs, Andrew D Mellinger, and John L Gallup. The geography of poverty and wealth. *Scientific American*, 284(3):70–75, 2001.

[263] Satya S. Sahoo, Wolfgang Halb, Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau Jr, Sören Auer, Juan Sequeda, and Ahmed Ezzat. A survey of current approaches for mapping of relational databases to rdf, 01 2009.

[264] Simon Scerri, Keith Cortis, Ismael Rivera, and Siegfried Handschuh. Knowledge discovery in distributed social web sharing activities. *Making Sense of Microposts (# MSM2012)*, pages 26–33, 2012.

[265] Benjamin Schäfer, Petar Ristoski, and Heiko Paulheim. What is special about bethlehem, pennsylvania? identifying unexpected facts about dbpedia entities. In *International Semantic Web Conferences, Posters and Demos*, 2015.

[266] Johann Schaible, Zeljko Carevic, Oliver Hopt, and Benjamin Zapilko. Utilizing the open movie database api for predicting the review class of movies.

In *4th International Workshop on Knowledge Discovery and Data Mining Meets Linked Open Data*, 2015.

[267] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. Adoption of the linked data best practices in different topical domains. In *The Semantic Web–ISWC*. 2014.

[268] Max Schmachtenberg, Thorsten Strufe, and Heiko Paulheim. Enhancing a location-based recommendation system by enrichment with structured data from the web. In *Web Intelligence, Mining and Semantics*, 2014.

[269] Michael Schuhmacher and Simone Paolo Ponzetto. Exploiting dbpedia for web search results clustering. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 91–96. ACM, 2013.

[270] Michael Schuhmacher and Simone Paolo Ponzetto. Knowledge-based graph document modeling. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 543–552. ACM, 2014.

[271] Axel Schulz, Christian Guckelsberger, and Frederik Janssen. Semantic abstraction for generalization of tweet classification.

[272] Axel Schulz and Frederik Janssen. What is good for one city may not be good for another one: Evaluating generalization for tweet classification based on semantic abstraction. In CEUR, editor, *Proceedings of the Fifth Workshop on Semantics for Smarter Cities a Workshop at the 13th International Semantic Web Conference*, volume 1280, pages 53–67, 2014.

[273] Axel Schulz, Petar Ristoski, and Heiko Paulheim. I see a car crash: Real-time detection of small scale incidents in microblogs. In *The Semantic Web: ESWC 2013 Satellite Events*, pages 22–33. Springer, 2013.

[274] J. Seitner, C. Bizer, K. Eckert, S. Faralli, R. Meusel, H. Paulheim, and S. Ponzetto. A large database of hypernymy relations extracted from the web. In *LREC*, 2016.

[275] Giovanni Semeraro, Pasquale Lops, Pierpaolo Basile, and Marco de Gemmis. Knowledge infusion into content-based recommender systems. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 301–304, New York, NY, USA, 2009. ACM.

[276] Floarea Serban, Joaquin Vanschoren, Jörg-Uwe Kietz, and Abraham Bernstein. A survey of intelligent assistants for data analysis. *ACM Computing Surveys (CSUR)*, 45(3):31, 2013.

[277] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *The Journal of Machine Learning Research*, 12:2539–2561, 2011.

[278] GURUH FAJAR SHIDIK and AHMAD ASHARI. Linked open government data as background knowledge in predicting forest fire. *Jurnal Informatika*, 2014.

[279] Carlos N. Silla, Jr. and Alex A. Freitas. A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov.*, pages 31–72, 2011.

[280] J Sivakumar et al. A review on semantic-based web mining and its applications. 2013.

[281] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013.

[282] Geir Solskinnsbakk and Jon Atle Gulla. Semantic annotation from social data. In *Proceedings of the Fourth International Workshop on Social Data on the Web Workshop*, 2011.

[283] Pascal Soucy and Guy W. Mineau. Beyond tfidf weighting for text categorization in the vector space model. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 1130–1135, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[284] Dimitrios-Emmanuel Spanos, Periklis Stavrou, and Nikolas Mitrou. Bringing relational databases into the semantic web: A survey. *Semant. web*, 3(2):169–209, April 2012.

[285] K Sridevi and Dr R UmaRani. A survey of semantic based solutions to web mining. *International Journal of Emerging Trends and Technology in Computer Science (IJETTS)*, 1, 2012.

[286] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In *VLDB*, volume 95, pages 407–419, 1995.

[287] S. Sritha and B. Mathumathi. A survey on various approaches for taxonomy construction. *Indian Journal of Innovations and Developments*, 5(6), 2016.

[288] Harald Steck. Evaluation of recommendations: Rating-prediction and ranking. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 213–220, New York, NY, USA, 2013. ACM.

[289] Gerd Stumme, Andreas Hotho, and Bettina Berendt. Semantic web mining - state of the art and future directions. *Journal of Web Semantics*, 4(2):124–143, 2006.

[290] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. PARIS: Probabilistic Alignment of Relations, Instances, and Schema. *PVLDB*, 5(3):157–168, 2011.

[291] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *16th international conference on WWW*, pages 697–706, 2007.

[292] Akihiro Suyama, Naoya Negishi, and Takahira Yamaguchi. Camlet: A platform for automatic composition of inductive learning systems using ontologies. In *PRICAI'98: Topics in Artificial Intelligence*, pages 205–215. Springer, 1998.

[293] Vojtěch Svátek, Jan Rauch, and Martin Ralbovská. Ontology-enhanced association mining. In Markus Ackermann, Bettina Berendt, Marko Grobelnik, Andreas Hotho, Dunja Mladenič, Giovanni Semeraro, Myra Spiliopoulou, Gerd Stumme, Vojtěch Svátek, and Maarten van Someren, editors, *Semantics, Web and Mining*, volume 4289 of *Lecture Notes in Computer Science*, pages 163–179. Springer Berlin Heidelberg, 2006.

[294] Zareen Syed, Tim Finin, Varish Mulwad, and Anupam Joshi. Exploiting a web of semantic data for interpreting tables. In *In: Proceedings of the Second Web Science Conference.*, 2010.

[295] Andreas Thalhammer. Dbpedia pagerank dataset. Downloaded from `http://people.aifb.kit.edu/ath/#DBpedia_PageRank`, 2014.

[296] Andreas Thalhammer and Achim Rettinger. PageRank on Wikipedia: Towards General Importance Scores for Entities. In *The Semantic Web: ESWC 2016 Satellite Events*, pages 227–240. Springer International Publishing, Crete, Greece, 2016.

[297] Rajesh Thiagarajan, Geetha Manjunath, and Markus Stumptner. Computing semantic similarity using ontologies. *HP Laboratories). Technical report HPL-2008-87*, 2008.

[298] Ilaria Tiddi. Explaining data patterns using background knowledge from linked data, 2013.

[299] Ilaria Tiddi, Mathieu d'Aquin, and Enrico Motta. Explaining clusters with inductive logic programming and linked data. In *Proceedings of the ISWC 2013 Posters & Demonstrations Track, Sydney, Australia, October 23, 2013*, pages 257–260, 2013.

[300] Ilaria Tiddi, Mathieu d'Aquin, and Enrico Motta. Dedalo: Looking for clusters explanations in a labyrinth of linked data. In Valentina Presutti, Claudia d'Amato, Fabien Gandon, Mathieu d'Aquin, Steffen Staab, and Anna Tordai, editors, *The Semantic Web: Trends and Challenges*, volume 8465 of *Lecture Notes in Computer Science*, pages 333–348. Springer International Publishing, 2014.

[301] Ilaria Tiddi, Mathieu d'Aquin, and Enrico Motta. Using neural networks to aggregate linked data rules. In Krzysztof Janowicz, Stefan Schlobach, Patrick Lambrix, and Eero Hyvönen, editors, *Knowledge Engineering and Knowledge Management*, volume 8876 of *Lecture Notes in Computer Science*, pages 547–562. Springer International Publishing, 2014.

[302] Ilaria Tiddi, Mathieu d'Aquin, and Enrico Motta. Walking linked data: a graph traversal approach to explain clusters. In *Proceedings of the 5th International Workshop on Consuming Linked Data (COLD 2014) co-located with the 13th International Semantic Web Conference (ISWC 2014), Riva del Garda, Italy, October 20, 2014.*, 2014.

[303] Binyam Tilahun, Tomi Kauppinen, Carsten Keßler, and Fleur Fritz. Design and development of a linked open data-based health information representation and visualization system: Potentials and preliminary evaluation. *JMIR medical informatics*, 2(2), 2014.

[304] K. M. Ting and I. H. Witten. Issues in stacked generalization. *Artificial Intelligence Research*, 10(1), 1999.

[305] Igor Trajkovski, Nada Lavrač, and Jakub Tolar. Segs: Search for enriched gene sets in microarray data. *Journal of biomedical informatics*, 41(4):588–601, 2008.

[306] Volker Tresp, Markus Bundschus, Achim Rettinger, and Yi Huang. Uncertainty reasoning for the semantic web i. chapter Towards Machine Learning on the Semantic Web, pages 282–314. Springer-Verlag, Berlin, Heidelberg, 2008.

[307] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080, 2016.

[308] Giovanni Tummarello, Richard Cyganiak, Michele Catasta, Szymon Danielczyk, Renaud Delbru, and Stefan Decker. Sig. ma: Live views on the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4):355–364, 2010.

[309] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[310] Jörg Unbehauen, Sebastian Hellmann, Sören Auer, and Claus Stadler. Knowledge extraction from structured sources. In Stefano Ceri and Marco Brambilla, editors, *Search Computing*, volume 7538 of *Lecture Notes in Computer Science*, pages 34–52. Springer Berlin Heidelberg, 2012.

[311] Marieke van Erp, Pablo Mendes, Heiko Paulheim, Filip Ilievski, Julien Plu, Giuseppe Rizzo, and Jörg Waitelonis. Evaluating entity linking: An analysis of current benchmark datasets and a roadmap for doing a better job. In *10th International Conference on Language Resources and Evaluation (LREC)*, 2016.

[312] WillemRobert van Hage, Marieke van Erp, and Véronique Malaisé. Linked open piracy: A story about e-science, linked data, and statistics. *Journal on Data Semantics*, 1(3):187–201, 2012.

[313] Anze Vavpetič and Nada Lavrač. Semantic subgroup discovery systems and workflows in the sdm-toolkit. *Comput. J.*, pages 304–320, 2013.

[314] Anže Vavpetič, Petra Kralj Novak, Miha Grčar, Igor Mozetič, and Nada Lavrač. Semantic data mining of financial news articles. In Johannes Fürnkranz, Eyke Hüllermeier, and Tomoyuki Higuchi, editors, *Discovery Science*, volume 8140 of *Lecture Notes in Computer Science*, pages 294–307. Springer Berlin Heidelberg, 2013.

[315] Anže Vavpetič, Vid Podpečan, and Nada Lavrač. Semantic subgroup explanations. *Journal of Intelligent Information Systems*, 42(2):233–254, 2014.

[316] P. Velardi, S. Faralli, and R. Navigli. OntoLearn Reloaded: A Graph-Based Algorithm for Taxonomy Induction. *Computational Linguistics*, 39(3):665–707, 2013.

[317] Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Paşca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. Recovering semantics of tables on the web. *Proc. VLDB Endow.*, 4(9):528–538, June 2011.

[318] Johanna Völker and Mathias Niepert. Statistical schema induction. In *Extended Semantic Web Conference*, pages 124–138. Springer, 2011.

[319] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[320] Bill B. Wang, R. I. Bob Mckay, Hussein A. Abbass, and Michael Barlow. A comparative study for domain ontology guided feature extraction. In *Australasian Computer Science Conference*, 2003.

[321] Jingjing Wang, Haixun Wang, Zhongyuan Wang, and Kenny Q. Zhu. Understanding tables on the web. In *Proceedings of the 31st International Conference on Conceptual Modeling*, ER'12, pages 141–155, Berlin, Heidelberg, 2012. Springer-Verlag.

[322] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 2017.

[323] Xin Wang, Howard John Hamilton, and Yashu Bither. *An ontology-based approach to data cleaning*. Regina: Department of Computer Science, University of Regina, 2005.

[324] Yongheng Wang and Shenghong Yang. Outlier detection from massive short documents using domain ontology. In *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*, volume 3, pages 558–562. IEEE, 2010.

[325] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119. Citeseer, 2014.

[326] Chris Webb. *Power Query for Power BI and Excel*. Springer, 2014.

[327] Scott White and Padhraic Smyth. Algorithms for estimating relative importance in networks. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 266–275, New York, NY, USA, 2003. ACM.

[328] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q. Zhu. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 481–492, New York, NY, USA, 2012. ACM.

[329] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374. ACM, 2015.

[330] Bushra Zafar, Michael Cochez, and Usman Qamar. Using distributional semantics for automatic taxonomy induction. In *Frontiers of Information Technology (FIT), 2016 International Conference on*, pages 348–353. IEEE, 2016.

[331] Monika Žáková, Petr Kremen, Filip Zelezny, and Nada Lavrac. Planning to learn with a knowledge discovery ontology. In *SECOND PLANNING TO LEARN WORKSHOP (PLANLEARN) AT THE ICML/COLT/UAI 2008*, page 29, 2008.

[332] Monika Žáková, Petr Kremen, Filip Zelezny, and Nada Lavrac. Automating knowledge discovery workflow composition through ontology-based planning. *Automation Science and Engineering, IEEE Transactions on*, 8(2):253–264, 2010.

[333] Monika Záková, Vid Podpecan, Filip Zeleznỳ, and Nada Lavrac. Advancing data mining workflow construction: A framework and cases using the orange toolkit. *Proc. 2nd Intl. Wshop. Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery*, pages 39–52, 2009.

[334] Martin Zeman, Martin Ralbovskỳ, Vojtech Svátek, and Jan Rauch. Ontology-driven data preparation for association mining. *Online http://keg. vse. cz/onto-kdd-draft. pdf*, 2009.

[335] Meihui Zhang and Kaushik Chakrabarti. InfoGather+: Semantic Matching and Annotation of Numeric and Time-varying Attributes in Web Tables. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 145–156, New York, NY, USA, 2013. ACM.

[336] X. Zhang, Q. Yuan, S. Zhao, W. Fan, W. Zheng, and Z. Wang. Multi-label classification without the multi-label cost. In *Proceedings of the 2010 SDM*, 2010.

[337] Ziqi Zhang. Learning with partial data for semantic table interpretation. In Krzysztof Janowicz, Stefan Schlobach, Patrick Lambrix, and Eero Hyvönen, editors, *Knowledge Engineering and Knowledge Management*, volume 8876 of *Lecture Notes in Computer Science*, pages 607–618. Springer International Publishing, 2014.

[338] Ziqi Zhang. Start small, build complete: Effective and efficient semantic table interpretation using tableminer. *Under transparent review: The Semantic Web Journal*, 2014.

[339] Ziqi Zhang, Anna Lisa Gentile, and Isabelle Augenstein. "linked data as background knowledge for information extraction on the web". *SIGWEB Newsl.*, (Summer):5:1–5:9, July 2014.

[340] Xuan Zhou and James Geller. Raising, to enhance rule mining in web marketing with the use of an ontology. *Data Mining with Ontologies: Implementations, Findings and Frameworks*, pages 18–36, 2007.