# Knowledge-based Graph Document Modeling

Michael Schuhmacher
Data and Web Science Research Group
University of Mannheim, Germany
michael@informatik.uni-mannheim.de

Simone Paolo Ponzetto
Data and Web Science Research Group
University of Mannheim, Germany
simone@informatik.uni-mannheim.de

## ABSTRACT

We propose a graph-based semantic model for representing document content. Our method relies on the use of a semantic network, namely the DBpedia knowledge base, for acquiring fine-grained information about entities and their semantic relations, thus resulting in a knowledge-rich document model. We demonstrate the benefits of these semantic representations in two tasks: entity ranking and computing document semantic similarity. To this end, we couple DBpedia's structure with an information-theoretic measure of concept association, based on its explicit semantic relations, and compute semantic similarity using a Graph Edit Distance based measure, which finds the optimal matching between the documents' entities using the Hungarian method. Experimental results show that our general model outperforms baselines built on top of traditional methods, and achieves a performance close to that of highly specialized methods that have been tuned to these specific tasks.

## Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; I.2.4 [**Artificial Intelligence**]: Semantic Networks; I.2.7 [**Artificial Intelligence**]: Natural Language Processing

## Keywords

Document modeling; Semantic network mining; DBpedia; Entity relatedness; Document semantic similarity.

## 1. INTRODUCTION

Recent years have seen a great deal of work on developing wide-coverage semantic technologies and methods embedding semantic models within a wide spectrum of applications, crucially including end-user applications like, for instance, question answering [16, 47], document search [14] and web-search results clustering [34]. Complementary to this trend, many research efforts have concentrated on the automatic acquisition of machine-readable knowledge on a large scale by mining large repositories of textual data such as the Web [2, 9] (inter alia), and exploiting collaboratively-constructed resources [40, 36, 21, 25]. As a result of this, recent years have seen a remarkable renaissance of knowledge-rich approaches for many different Natural Language Processing (NLP) and Information Retrieval (IR) tasks [27].

But while recent research trends indicate that semantic information and knowledge-rich approaches can be used effectively for high-end IR and NLP tasks, much still remains to be done in order to effectively exploit these rich models and further advance the state of the art in these fields. Most of the approaches which draw upon document representations, in fact, rely solely on morpho-syntactic information by means of 'flat' meaning representations like vector space models [44]. Although more sophisticated models have been proposed – including conceptual [17] and grounded [8] vector spaces – these still do not exploit the relational knowledge and network structure encoded within wide-coverage knowledge bases such as YAGO [25] or DBpedia [6].

In this paper, we aim at overcoming these issues by means of a knowledge-rich method to represent documents in the Web of Linked Data. Key to our approach is the combination of a fine-grained relation vocabulary with information-theoretic measures of concept associativity to produce a graph-based interpretation of texts leveraging large amounts of structured knowledge, i.e., disambiguated entities and explicit semantic relations, encoded within DBpedia. Our contributions are as follows:

- We propose a graph-based document model and present a method to produce structured representations of texts that combine disambiguated entities with fine-grained semantic relations;
- We present a variety of information-theoretic measures to weight different semantic relations within an ontology, and automatically quantify their degree of relevance with respect to the concepts they connect. Edges in the semantic graphs are thus weighted so as to capture the degree of associativity between concepts, as well as their different levels of specificity;
- We evaluate our model using two highly relevant tasks, namely entity ranking and computing document similarity. We show that our approach not only outperforms standard baselines relying on traditional, i.e., 'flat', document representations, but also produces results close to those of highly specialized methods that have been particularly tuned to the respective tasks.

- We develop a new measure, based on graph edit distance techniques, in order to compute document similarity using our semantic graphs. Our approach views computing semantic distances within an ontology as a concept matching problem, and uses the Hungarian method for solving this combinatorial optimization problem.

As a result of this, we are able to provide a complete framework where documents are semantified by linking them to a reference knowledge base, and subgraphs of the knowledge resource are used for complex language understanding tasks like entity ranking and document semantic similarity. Results on entity ranking show that our weighting scheme helps us better estimate entity relatedness when compared with using simple, unweighted paths. Moreover, by complementing large amounts of knowledge with structured text representations we are able to achieve a robust performance on the task of computing document semantic similarity, thus competing with 'flat' approaches based on either word or conceptual vector spaces, while at the same time providing a general, *de-facto* parameter-free model.

## 2. MODELING DOCUMENT CONTENT WITH ONTOLOGY-BASED GRAPHS

We present our method to generate structured representations of textual content using DBpedia as the backend ontology. To this end, we opt for graph-based methods since: (i) they are general in nature, and can be used with *any* knowledge graph, i.e., a knowledge resource that can be viewed as a graph, regardless of its specific vocabulary; (ii) they have been shown to be effective for language understanding tasks when combined with labeled and unlabeled resources (cf., e.g., [36, 26]). In the following, we first briefly introduce DBpedia, the resource used in our methodology, in Section 2.1. Sections 2.2 through 2.3 illustrate instead the main phases of our approach.

### 2.1 DBpedia

Our approach relies on the information and structure encoded within an underlying knowledge base. In this work, we opt for DBpedia [6], since it provides a wide-coverage knowledge base with many (i.e., more than 1000) fine-grained explicit semantic relations between entities. However, our method can be also used with any other lexical or ontological resource, e.g. YAGO [25], provided it can be cast as a knowledge graph containing disambiguated entities and explicit semantic relations.

DBpedia is a community effort to extract structured information from Wikipedia and make this information available on the Web as a full-fledged ontology. The key idea behind DBpedia is to parse *infoboxes*, namely property-summarizing tables found within Wikipedia pages, in order to automatically acquire properties and relations about a large amount of entities. These are further embedded within an ontology based on Semantic Web formalisms like: i) representing data on the basis of the best practices of *linked data* [5]; ii) encoding semantic relations using the resource description framework (RDF), a generic graph-based data model for describing objects and their relationships. In the following, we refer to RDF triples in DBpedia consisting of a subject, predicate and object as a directed relation from *Subj* to *Obj*, connected by the labeled relation *Pred*.

Naturally, we can view DBpedia as a graph, whose nodes are entities and edges capture explicit semantic relations between them. For instance, an entity like "Bob Dylan" is connected in DBpedia to other entities by means of statements such as `db:Bob_Dylan rdf:type dbo:MusicalArtist` or `db:Bob_Dylan dbo:genre db:Folk_rock`, and so on[1]. We now show how we can use this structure to produce graph-based representations from an input consisting of sets of entities such as those mentioned in natural language texts.

### 2.2 Semantic graph construction

Let $C_{db}$ be the full set of DBpedia's concepts and entities[2] and $C$ an arbitrary subset of it, given as input – e.g., the set of entities mentioned within a document. In the first phase of our methodology, we create from the set of input entities a labeled, directed graph $G = (V, E)$ containing i) the entities themselves, ii) their semantic relations, as well as iii) any additional entity that is related to any of the input ones by means of some relation in the ontology. That is, $C \subseteq V \subseteq C_{db}$ and $E \subseteq V \times R \times V$, where $r \in R$ is a semantic relation found in DBpedia, e.g., `rdf:type`, `dbo:birthDate` or `dbp:genre` (we do not make any distinction between *A-box* and *T-box* statements, since we remain agnostic as to the specific vocabulary used by our underlying resource). Additionally, we want to associate a weight $w$ with each edge $(v_i, r, v_j) \in E$, in order to capture the degree of associativity between the source and target nodes – i.e., how strongly related the two corresponding entities are.

To produce our semantic graphs, we start with a set of input entities $C$ and create a labeled directed graph $G = (V, E)$ as follows: a) first, we define the set of nodes $V$ of $G$ to be made up of all input concepts, that is, we set $V := C$; b) next, we connect the nodes in $V$ based on the paths found between them in DBpedia. Nodes in $V$ are expanded into a graph by performing a depth-first search along the DBpedia graph and successively adding all outgoing relations[3] $r$, thus adding all simple directed paths $v, v_1, \ldots, v_k, v'$ of maximal length $L$ that connect them to $G$, i.e., $V := V \cup \{v_1, \ldots, v_k\}$, $E := E \cup \{(v, r_1, v_1), \ldots, (v_k, r_k, v')\}$. As a result, we obtain a sub-graph of DBpedia containing the initial concepts, together with all edges and intermediate concepts found along all paths of maximal length $L$ that connect them. In this work, we set $L = 2$ following a large body of evidence from previous related work [36, 28].

Figure 1 illustrates an example of a semantic graph generated from the set of entities { `db:Bob_Dylan`, `db:Monterey_Country_Fairgrounds`, `db:Mozambique_(Song)`, `db:Johnny_Cash` }, e.g. as found within the sentence "Dylan played Mozambique at Monterey right before Cash". Starting from these seed entities, we perform a depth-first search to add relevant intermediates concepts and relations to $G$ (e.g., `foaf:Person` or `db:Folk_music`). As a result of this, we obtain a semantically-rich graph: additional nodes and edges provide us with a rich structured context, in which the initial concepts are now connected by a variety of entities and explicit semantic relations.

[1]We abbreviate URI namespaces with common prefixes, see `http://prefix.cc` for details.
[2]Hereafter, we use *concept* and *entity* interchangeably to refer to resources of the knowledge base, i.e., DBpedia URIs.
[3]We filter out any administrative information and data using a list of stop-URIs provided by [28] and extended by us.

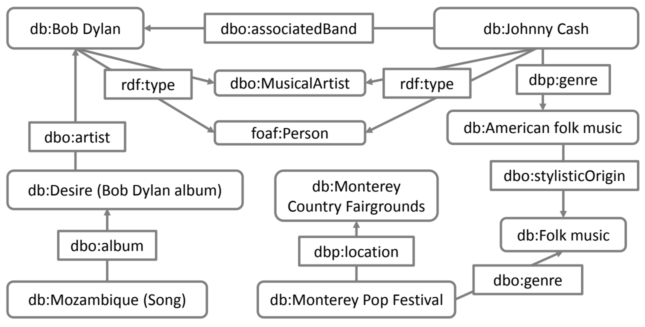

**Figure 1: Sample semantic graph.**

## 2.3 Semantic relation weighting

So far, our method simply connected a set of input entities by traversing the DBpedia graph – i.e., similar in spirit to graph-based approaches to Word Sense Disambiguation using lexical resources [36]. However, in contrast to lexical resources like WordNet, our backend ontology contains many different, fine-grained semantic relations: accordingly, not all relations are equally informative and questions arise about what kind of information to take into account when building the semantic graphs. For instance, in our example there exists multiple paths between the source nodes `db:Bob_Dylan` and `db:Johnny_Cash`, due to the typical high density of the DBpedia graph. Connecting paths, however, include highly informative relations (e.g., the two entities being linked directly via `dbo:associatedBand`), as well as generic ones (both being entities of `rdf:type foaf:Person`), which tend to apply to a very large amount of entities (i.e., all persons in DBpedia) and thus carry low discriminative power – e.g., in order to identify relations useful for computing semantic similarity (see Section 4). But while previous work [28] restricted the semantic relations used to build semantic graphs to a small, manually-selected set, we opt here instead for an automatic approach based on relation-specific edge weighting. This is because, while a manual approach ensures overall good quality, it does not scale and needs to be tuned for every knowledge base in turn. Consequently, we extend our semantic graphs by weighting their edges. Weights are meant to capture the degree of associativity between concepts in the graph – i.e., the degree of relevance of an edge (i.e., semantic relation) for the entities it connects. The key idea underlying our weighting is to reward, for a given source node, those edges and target nodes that are most specific to it. At the core of our edge weighting lies the notion of Information Content ($IC$):

$$IC_{X_{Pred}}\left(\omega_{Pred}\right) = -\log\left(P\left(\omega_{Pred}\right)\right) ,$$

where $P\left(\omega_{Pred}\right)$ is the probability that the random variable $X_{Pred}$ describing the type of edge, i.e. a specific semantic relation, shows the outcome $\omega_{Pred}$. This measure makes the assumption that specificity is a good proxy for relevance – cf., for instance the `rdf:type` vs. `dbo:associatedBand` predicates. We can compute these $IC$ values for all types of predicates, as we have the full DBpedia graph available and can query for all potential realizations of the random variable $X_{Pred}$. In our example, an edge labeled with `rdf:type` will accordingly get an $IC$ which is comparably lower than, say, one labeled with `dbo:associatedBand`.

**Joint Information Content (jointIC)**. While the Information Content of semantic relations provides us with a way to distinguish general vs. specific connections, it only covers the *a-priori* specificity of an edge, i.e., regardless of the entities it actually connects. However, as shown in Figure 1, the same type of edge, e.g. `rdf:type`, can lead to very general concepts with low discriminative power (`foaf:Person`), but also to very informative (because rare) ones, like `dbo:MusicalArtist`, which do, in fact, provide valuable information. We capture this by adding the conditional information content $IC\left(\omega_{Obj}|\omega_{Pred}\right)$ to our weighting scheme, which accounts for the concept the predicate is pointing to, given that the edge has already been observed. Formally, given an edge $e = (Subj, Pred, Obj)$ we compute the information content of the joint probability distribution, $IC\left(\omega_{Pred}, \omega_{Obj}\right)$, which we take as our weighting function:

$$w_{jointIC}\left(e\right) = IC\left(\omega_{Pred}\right) + IC\left(\omega_{Obj}|\omega_{Pred}\right) .$$

In our example, the `rdf:type` edge leading to `dbo:Musical Artist` accordingly receives a much higher weight than that pointing to the far more generic `foaf:Person`.

**Combined Information Content (combIC)**. Joint information content, although taking into account predicate and object specificity at the same time, can nevertheless penalize infrequent objects that occur with infrequent predicates – e.g., `db:American_folk_music` being overall very infrequent, but getting a high probability (and, hence, a low $IC$) when occurring conditional on `dbo:genre`. We propose to mitigate this problem by computing the joint information content while making an independence assumption between the predicated and the object. The resulting weights are then computed as the sum of the Information Content of the predicate and the object:

$$w_{combIC}\left(e\right) = IC\left(\omega_{Pred}\right) + IC\left(\omega_{Obj}\right) .$$

**Information Content and Pointwise Mutual Information (IC+PMI)**. An alternative way to compute the strength of association between the predicate and the object is by means of Pointwise Mutual Information (PMI):

$$PMI\left(\omega_{Pred}, \omega_{Obj}\right) = \log \frac{P\left(\omega_{Pred}, \omega_{Obj}\right)}{P\left(\omega_{Pred}\right) P\left(\omega_{Obj}\right)} .$$

PMI measures the mutual dependence between the two variable outcomes $\omega_{Pred}$ and $\omega_{Obj}$, and can thus be seen as a measure of how much deviation from independence there is between the two outcomes, i.e., the specific predicate and object found along a DBpedia graph edge. Our hunch here is to use PMI to find a middle ground between the assumption of full dependence (jointIC) or independence (combIC) between predicates and objects. We additionally combine PMI with the IC of the predicate, in order to bias our weights towards less frequent, and thus more informative, predicates:

$$w_{IC+PMI}\left(e\right) = IC\left(\omega_{Pred}\right) + PMI\left(\omega_{Pred}, \omega_{Obj}\right) .$$

## 3. GRAPH-BASED ENTITY RANKING

We present an extrinsic evaluation of the different weighting schemes we developed in order to build weighted semantic graphs from DBpedia. This is because semantic relation weighting makes a contribution of its own, and it can be

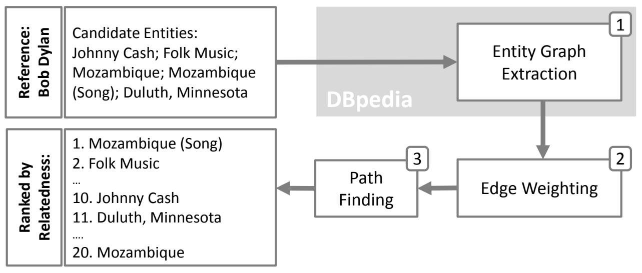

**Figure 2: Entity ranking workflow.**

used for a variety of tasks other than semantic graph construction – e.g., RDF triple ranking [23]. Consequently, we explore here a task-based evaluation on ranking related entities [24] in a knowledge base.

**Task description.** Entity ranking [12] is the task of ordering a given set of entities on the basis of their relevance with respect to a specific reference entity. In our case, since we work with DBpedia as knowledge base, we take, e.g., `db:Bob_Dylan` as reference and try to compute, how strongly `db:Johnny_Cash` is related to it, in comparison to `db:Folk_music` or `db:Mozambique_(Song)`, and so on. This ranking task has the advantage that it provides a focused, extrinsic evaluation of our different weighting methods: besides, there exists established gold standard datasets against which we can compare our approach. Entity ranking can be seen as similar in spirit to computing word relatedness [49], except that in our setting we are given as input unambiguous entity references, rather than potentially ambiguous words. Besides, entity ranking also plays a key role in Entity Linking (EL) [10, 29], since EL relies on estimating the degree of relatedness between candidate entity references of different mentions in text. That is, within a global document-level EL approach, entity mentions can be jointly disambiguated by maximizing their degree of semantic overlap as obtained, for instance, from information stored within a knowledge base – cf. AIDA [26].

**Entity ranking method.** We present an overview of our approach in Figure 2. Given a reference entity *ref* and a set of entities $E_{Cand}$ (both found in DBpedia), our method ranks these entities by performing three main steps:

1) we build a semantic graph following the procedure of Section 2.2 using all candidate entities $E_{Cand}$, as well as the reference instance as input concepts;

2) we weight graph edges by edge cost, which is defined as

$$c(e) = w_{max} - w(e) \; , \qquad (1)$$

where $w(.)$ is any of the three weighting functions defined in Section 2.3, and $w_{max}$ is the globally highest possible weight in the DBpedia graph for the selected weighting function;

3) finally, we compute (weighted) semantic distances between entity pairs – i.e., the reference entity *ref* and each of the entities *cand* in $E_{Cand}$ in turn – as the minimum path cost between them in our weighted graph:

$$distance(ref, cand) = \min_{p \in paths(ref,cand)} c_p(ref, cand) \, .$$

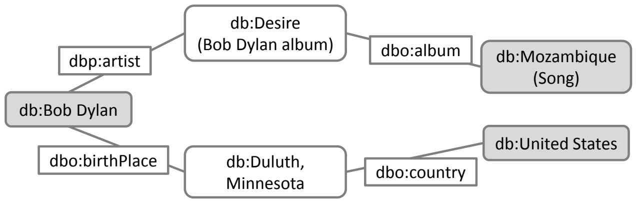

**Figure 3: Example of multiple paths between entities with different semantic specificity.**

| | Unwghtd | jointIC | combIC | IC+PMI |
|---|---|---|---|---|
| Hollywood Celebr. | 0.639 | 0.541 | **0.690** | 0.661 |
| IT Companies | 0.559 | 0.636 | **0.644** | 0.583 |
| Television Series | 0.529 | 0.595 | **0.643** | 0.602 |
| Video Games | 0.451 | **0.562** | 0.532 | 0.484 |
| Chuck Norris | 0.458 | 0.409 | **0.558** | 0.506 |
| All 21 Entities | 0.541 | 0.575 | **0.624** | 0.579 |

**Table 1: Performance on the entity ranking KORE dataset (best results are bolded).**

We then rank the entity pairs increasingly by semantic distance. Each single path cost between two entities is calculated as the sum of the edge costs along their undirected connecting path $p$:

$$c_p(ref, cand) = \sum_{e \in \{(ref,r_1,v_1),\ldots,(v_k,r_k,cand)\}} c(e) \, . \qquad (2)$$

As a result of our method, we are able to determine the degree of relatedness between two arbitrary instances within our background knowledge base as the inverse of their semantic distance. We briefly illustrate our method in Figure 3 with an example using `db:Bob_Dylan` and `db:Mozambique_(Song)` as input entity pair. When looking at the entity `db:Bob_Dylan` (the musician), we note that it is not directly connected to his song, `db:Mozambique_(Song)`. However, thanks to the fact that DBpedia encodes very specific facts – namely i) that Bob Dylan is the main artist of the album `db:Desire_(Bob_Dylan_album)`, and ii) that `db:Mozambique_(Song)` is a song contained in that very same album – we are able to estimate a high degree of semantic relatedness between the two input entities.

Note that our weighting scheme plays a crucial role in estimating the degree of semantic overlap. If we look, for instance, to another entity pair such as the one consisting of `db:Bob_Dylan` and `db:United_States`, we note that in DBpedia these entities are connected by a short, albeit rather uninformative (because unspecific), path consisting of a single intermediate entity (`db:Duluth,_Minnesota`). Our weighting captures this by assigning a low weight to edges denoting general semantic relations such as `dbo:birthPlace` and `dbo:country`. As a result of this, we are able to rank `db:Mozambique_(Song)` higher than `db:United_States`, although both are connected to the reference entity `db:Bob_Dylan` by a path of equal length.

**Experimental setting and evaluation.** We use the KORE entity ranking dataset from Hoffart et al. [24]. The dataset consists of 21 different reference entities from four different domains, namely IT companies, Hollywood celebrities, television series, video games, and Chuck Norris (a singleton dataset). For each ranking problem, Hoffart et al.

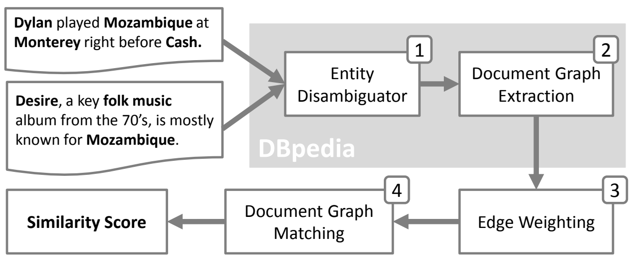
Dylan played Mozambique at Monterey right before Cash.
Desire, a key folk music album from the 70's, is mostly known for Mozambique.
DBpedia
1
Entity Disambiguator
2
Document Graph Extraction
3
Edge Weighting
4
Document Graph Matching
Similarity Score

**Figure 4: Document semantic similarity workflow.**

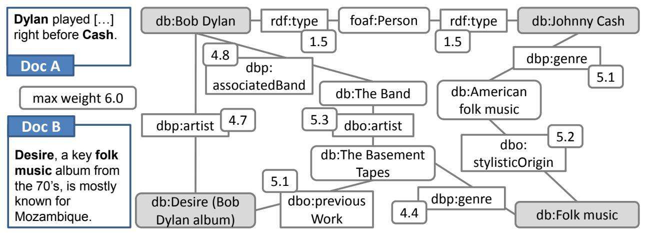

**Figure 5: Graph document comparison (numbers on the graph edges indicate edge weights).**

selected a set of 20 candidate entities, which were found to be related with different degrees to the reference entity. Relatedness assessments were obtained from human judges using a crowd-sourcing approach. As an example, the entity "Apple Inc." (from the 'IT Companies' category) is paired with, among others, the following other entities:

| | |
|---|---|
| **Reference entity** | Apple Inc. |
| **Related entity (rank)** | Steve Jobs (1), Steve Wozniak (2), . . . Silicon Valley (9), NeXT (10) . . . Ford Motor Company (20) |

Naturally, different entities have different degrees of relatedness with the concept of "Apple" as a company. "Steve Jobs", for instance, ranks highest, having been a key figure of the company. In the middle range, instead, we find related companies such as "NeXT", another company founded by Steve Jobs (rank 10). Finally, at the end of the ranking we find "Ford Motor Company", which is only marginally related to "Apple", being also an American company but from a completely different industry.

We follow the original evaluation setting of Hoffart et al. and compute Spearman's rank correlation coefficient $\rho$ for each reference entity in turn. Overall results are then obtained by averaging over all reference entities in the dataset. We report our results in Table 1, where we compare our different weighting schemes from Section 2.3. As baseline we use an unweighted version of the DBpedia graph: this amounts to computing entity relatedness simply as a function of distance in the network. Looking at the overall performance of the three alternative weighting schemes for all 21 ranking tasks, we observe that combIC consistently outperforms the baseline and both jointIC and IC+PMI on three domains out of four. When looking at specific domains, jointIC does not always improve the baseline, as results for Chuck Norris and Hollywood celebrities are actually getting worse. Nevertheless, on average all 3 weighting methods improve the baseline, with combIC, which shows an average increase of 15.5% (statistically significant for each task at $p \leq .001$ level using a paired t-test), achieving the best results. When compared with the original results from Hoffart et al. [24], our method achieves a performance slightly lower than their original proposal ($\rho = 0.673$), while at the same time outperforming all its approximations ($\rho = 0.621$ and 0.425). Overall, we take these performance figures to indicate the high quality of weighted connecting paths between entities in DBpedia. Consequently, we now take this idea one step further and use these paths to provide a structured representation of unstructured data, i.e. natural language texts. We then use these graph-based semantic representations to compute semantic similarity between documents.

## 4. COMPUTING DOCUMENT SIMILARITY

We present an application of our semantic graphs to the task of computing semantic similarity between texts. We provide an overview of our approach in Figure 4. Our method starts with the output of an entity disambiguator, which is used to identify a set of concepts from the input texts (1). Next, connecting paths between entities are collected, in order to identify the sub-graph of DBpedia covered by each document (2). Nodes in the semantic graph consist of concepts capturing the main topics of the documents: in addition, edges in the graph are weighted to identify the semantic relations that are most relevant for these concepts (3). Finally, we view computing semantic similarity as a matching problem between the concepts of different documents, and apply a Graph Edit Distance based similarity measure to identify the 'best' connecting paths between the documents' concepts (4). As a result, we are able to output the degree of similarity of the two input documents.

### 4.1 Document graph construction

Given an input text document, we first semantify it by identifying the set of concepts it contains. To this end, words and phrases are annotated with DBpedia concepts using a document entity linking system, e.g., DBpedia Spotlight [32]. Given a mention and its candidate entities, the entity linker finds its most likely meaning in context – e.g., like Spotlight, using a Vector Space Model (based on a bag-of-words approach). Accordingly, given a document, we are able to obtain a set of disambiguated entities associated with its words and phrases. In the two example documents of Figure 5, we extract, for instance, key concepts like `db:Bob_Dylan`, `db:Johnny_Cash` and `db:Desire_(Bob_Dylan_album)`. We call these extracted concepts the source nodes $V_s^d$ of a document graph $G^d = (V^d, E^d), V_s^d \subseteq V^d$ representing document $d$. $G^d$ is built by applying the procedure described in Section 2.2 while using $V_s^d$ as the set of input entities.

### 4.2 Graph-based document similarity

Since we represent documents as weighted DBpedia subgraphs (Section 2), we can naturally formulate computing document similarity as a graph matching problem. While there exist exact graph matching algorithms, e.g. based on graph isomorphism, we require our measure to be able to effectively quantify degrees of similarity. Consequently, we opt for an application of Graph Edit Distance techniques for our specific problem.

Graph Edit Distance (GED, [18]) is a general, inexact graph matching method that defines the distance between two graphs in terms of the minimum cost of edit operations needed to transform one graph into the other. In general, a GED measure needs to define edit cost functions for insertion, deletion, and modification for both nodes and edges. However, given our specific problem setting, we drop some of these requirements and define only cost functions for nodes. This is because, given a pair of semantic graphs, generated using the method from Section 2, these actually consist of two subgraphs of the same background model, namely DBpedia. As a result, no cost function over edges needs to be defined, since an edge existing or not in one graph will also be present or not in the other, given the fact that both document graphs belong in fact to the same supergraph. Thus, edit operation on edges solely cannot occur and, accordingly, we define edge cost functions to yield zero.

We define cost operations for nodes as follows. Note that, since we work with a well-defined ontology that represents concepts by unique URIs, we can rely on the fact that nodes in the DBpedia graph are unique. As a result, we do not need to account for label mismatch between concepts – e.g., the entity "Bob Dylan" being identified by `db:Robert_Allen_Zimmerman` in a graph, and referred to as `db:Bob_Dylan` in another one (or vice versa). Thus, in contrast to standard GED approaches, we define node modifications on the basis of the underlying edge structure, i.e., weighted distances in the graph, as opposed, for instance, to the application of string similarity measures like Levenshtein distance on node labels. The modification cost between two nodes is defined, analog to Section 3, as the sum of the edge costs along their connecting path (cf. Equation 2). By employing our edge cost function (Equation 1) we capture the fact that the closer (i.e., more semantically related) two nodes are, the lower the cost to modify one into the other is.

An exact solution to the GED problem can be found with a tree search over all possible edit operations, which, however, is computationally intractable for any reasonably-sized graph. In this work, we accordingly adapt an approximation method based on bipartite graph matching for finding the minimal edit cost [41]. This precomputes the cheapest node modification costs for each node pair first, and stores them into a cost matrix. Since in our case there can exist multiple paths between two nodes (and, thus, multiple such modification costs), we always select the cheapest node modification operation as the cheapest connecting path. Next, the matrix is extended with the cost for node insertion and deletion – which we define as equal to the most expensive node modification operation in the matrix (see below for details). Computing the GED is now a bipartite graph matching problem between the source nodes of the two graphs, with the objective of minimizing the edit cost and subject to the restriction of a strict one-to-one matching (as every node can only be modified exactly once). We solve this minimization problem using the Hungarian method (also known as Kuhn-Munkres or Munkres' algorithm). After computing the GED, we apply a simple normalization step to eliminate the effect of different graph, i.e., document sizes.

We summarize our approach in Algorithm 1. Given two semantic graphs $G^i$ and $G^j$, representing documents $d^i$ and $d^j$ (Section 2), we perform the following steps:

**Algorithm 1** Graph-based semantic similarity

```
Input: Document DBpedia subgraphs
       $G^i = (V^i, E^i)$, $G^j = (V^j, E^j)$
Parameter: Maximal path length $n_{max}$

 1: function SubgraphDistance($G^i, G^j$)
 2:     $P \leftarrow \emptyset$                              ▷ set of cheapest paths
 3:     for all $(v^i, v^j) \in V_s^i \times V_s^j$ from $G^i, G^j$ do
 4:         if $v^i = v^j$ then
 5:             $c^{i,j} \leftarrow 0$
 6:         else
 7:             $c^{i,j} \leftarrow \mathit{DijkstraCheapestPath}(v^i, v^j)$
 8:         $P \leftarrow P \cup \{(p^{i,j}, c^{i,j})\}$
 9:     $c^{max} \leftarrow \max_{p \in P_{length \leq n_{max}}}(c_p)$
10:     for all $(p^{i,j}, c^{i,j}) \in P$ do
11:         if $p^{i,j}_{length} \leq n_{max}$ then
12:             $c^{i,j} \leftarrow c^{i,j} / c^{max}$
13:         else
14:             $c^{i,j} \leftarrow 1$

15:     $D_m \leftarrow \{d_{i,j}\}_{i=1,\ldots,m,\ j=1,\ldots,m}, m = \max(|V_s^i|, |V_s^j|)$
16:     for all $d_{i,j}$ do
17:         if $i \leq j$ then                               ▷ Be $i \geq j$
18:             $d_{i,j} \leftarrow c^{i,j}$
19:         else
20:             $d_{i,j} \leftarrow c^{max}$
21:     $M \leftarrow \mathit{HungarianCheapestMatching}(D_m)$
22:     $\mathit{dist}(G^i, G^j) \leftarrow (\sum_{m \in M} m_{cost}) / |V_s^i \cup V_s^j|$
        return $\mathit{dist}(G^i, G^j)$
```

i) **lines 1–9**: for each pair of source nodes $V_s^i \times V_s^j$ we find the cheapest undirected path $p^{i,j}$ with cost $c^{i,j}$ using Dijkstra's algorithm (edges along the path are weighted by one of our three measures from Section 2.3). In our example in Figure 5, for instance, we compute the cheapest path between `db:Bob_Dylan` and `db:Johnny_Cash` from Doc A, and each of `db:Desire_(Bob_Dylan_album)` and `db:Folk_music` from Doc B in turn. The highest weighted edge, here `dbo:artist`, is assigned a cost of 0.7 (assuming a global upper edge cost limit $w_{max} = 6.0$, Equation 1), whereas the lowest weighted edge, namely the two `rdf:type` relations, are both assigned a cost of $6.0 - 1.5 = 4.5$. Given these costs, the cheapest path between, for instance, `db:Johnny_Cash` and `db:Folk_music` is the one through `db:American_folk_music`. Note that, in order to avoid long paths between very distant (and thus semantically unrelated) concepts, we limit the search based on a maximum search depth parameter $n_{max}$.

ii) **lines 10–14**: we next compute the node modification costs for each pair of source nodes. For paths found exceeding the path limit $n_{max}$, we set their cost to that of the most expensive path $c^{max}$ found within the input graph pair. Since it might not be the case that both graphs are fully connected, we also set $c^{max}$ as the cost for unconnected source node pairs. Finally, we normalize all cost values.

iii) **lines 15–20**: we build the final edit distance matrix $D_m$ from the previously computed modification costs, as well as the costs of the node insertion and deletion operations, which we set to $c^{max}$. This is to account for the fact that, given an arbitrary document pair, the cardinality of their sets of entities does not need to be the same: in this case, additional nodes are treated the same as very distant ones.

| | | | jointIC | combIC | IC+PMI |
|---|---|---|---|---|---|
| TagMe | Jaccard | | | | 0.51 |
| | GED | max depth @ 2 | 0.55 | 0.59 | 0.57 |
| | | @ 3 | 0.52 | 0.56 | 0.54 |
| | | @ 4 | 0.46 | 0.49 | 0.52 |
| Spotlight | Jaccard | | | | 0.54 |
| | GED | max depth @ 2 | **0.60** | **0.63** | **0.63** |
| | | @ 3 | 0.55 | 0.61 | 0.61 |
| | | @ 4 | 0.52 | 0.55 | 0.57 |
| | DKPro [3] | | | | 0.21 |
| | TakeLab [45] | | | | 0.08 |
| | Cosine baseline | | | | 0.56 |

**Table 2: Results on the LP50 dataset (Pearson $r$ correlation coefficient, best results are bolded).**

iv) **lines 21-22**: the edit distance matrix $D_m$ represents a bipartite matching problem, which we solve with the Hungarian method. This finds the optimal, cost-minimal assignment in our node operations matrix, while ensuring that each node will only be edited once. We finally normalize the graph edit distance costs to account for the number of source entities in the two input documents.

As a result of the execution of the algorithm, the normalized graph edit distance between $G^i$ and $G^j$ is returned. In our example, we will get a mapping from `db:Bob_Dylan` to `db:Desire_(Bob_Dylan_album)` (cost 1.3) and from `db:Johnny_Cash` to `db:Folk_music` (cost 0.9 + 0.8). The final similarity score is then given by the sum of these edit costs (3.0), normalized by the number of distinct source entities in both documents (6).

## 4.3 Experiments

**Experimental setting.** We evaluate our approach with a benchmarking dataset for document semantic similarity, in order to be able to compare our method against other state-of-the-art systems. To this end, we use the **LP50** dataset [31], 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, Lee et al. averaged for each pair the scores of all annotators: however, the final collection of 1,225 relatedness scores has only 67 distinct values. Consequently, Spearman's rank correlation is not appropriate to evaluate performance on this data and we opt instead, following previous work, for instance [17], for Pearson's linear correlation coefficient ($r$).

**Results and discussion.** We report our performance figures on the LP50 dataset in Table 2, where we show the Pearson product-moment correlation coefficient $r$ between the human gold standard and our graph-based approach (GED). In order to evaluate our method across different entity linking systems we test with both DBpedia Spotlight [32] and TagMe [15], two state-of-the-art systems [10]. For each tagger, we compute its performance with respect to different values for the maximum depth of the path search in the cost computation ($n_{\max}$). We compare our GED-based method with a variety of baselines:

i) a semantically-informed baseline which computes the Jaccard similarity coefficient over the set of entities identified within the input documents, namely $sim(d_1, d_2) = \frac{C^1 \cap C^2}{C^1 \cup C^2}$, where $C^1$ and $C^2$ represent the set of concepts identified by the entity tagger (i.e., TagMe or Spotlight) within documents $d_1$ and $d_2$, respectively;

ii) an unsupervised baseline computed as the cosine distance of a standard bag-of-words Vector Space Model;

iii) two strong supervised baselines based on two publicly available supervised systems, namely DKPro [3] and TakeLab [45], both trained on standard SemEval STS datasets.

In general, using our graph-based approach to document semantic similarity we are able to beat all baselines by a large margin, achieving a correlation coefficient of up to 0.63 ($n_{\max} = 2$, using Spotlight and either combIC or IC+PMI weighting). This is equal to a relative improvement of 16.0% over the semantically-informed Jaccard baseline and 11.6% over the cosine bag-of-words baseline[4]. The results indicate that our method is able to always perform above the Jaccard baseline for $n_{\max} \leq 3$, and achieves the best performance for $n_{\max} = 2$. These parameter values are indeed in-line with the optimal ones found by previous research contributions making use of graphs derived from Wikipedia or DBpedia [36, 28], which also showed the benefits of mining information from short, highly specific paths. This, in turn, makes our model virtually parameter-free, because it implies that we can simply set the only tunable parameter of our method, namely the depth of the search used for concept matching, to standard values (i.e., 2 or 3) which are known to yield good performance across many different tasks. When looking at the performance of the different weighting measures, we see that we consistently obtain the best results using either combIC or IC+PMI, which corroborates our previous findings on entity ranking (Section 3). Finally, we notice that the different baselines show large performance variations. The simple cosine baseline turns out to be a difficult competitor – e.g., outperforming the simple Jaccard baselines computed from both TagMe and Spotlight annotations – which indicates that semantifying the input texts and applying a simple entity overlap measure is not enough to yield a robust performance. The supervised baselines, DKPro and TakeLab, both show instead an extremely low performance rate, although they were reported as being among the top systems of the SemEval STS 2012 shared task. This is because both systems are supervised in nature, and thus able to yield accurate performance only when in-domain labeled data are available.

Next, in order to better understand the performance of our method, we compare it in Table 3 with an unweighted version that does not use edge weighting (i.e., all edge modifications have the same cost), as well as previous results from the literature. When computing semantic distances without weighting i.e., using the Hungarian method for mapping, but applied to unweighted paths only, we achieve up to $r = 0.61$ when using Spotlight and a maximum depth of 3 – 12.5%

[4]All differences in performance are statistically significant at $p < 0.05$ using Fisher's Z-value transformation unless otherwise noted.

|  | $r$ |
|---|---|
| GED-based (weighted) | 0.63 |
| GED-based (unweighted) | 0.61 |
| Bag-of-Words [31] | 0.1-0.5 |
| LSA [31] | 0.60 |
| ESA – original [17] | 0.72 |
| ESA – reimplemented [4] | 0.46-0.59 |

**Table 3: System comparison on the LP50 data.**

above the semantically-informed Jaccard baseline and 8.3% over the cosine bag-of-words baseline. This indicates the overall robustness of our GED method, which exploits high-quality semantic paths from DBpedia. Similar to our results on entity ranking, additional performance gains can be achieved thanks to weighting semantic relations.

When comparing our approach to the state of the art on this dataset we see that, while we outperform robust methods such as LSA [11], we are nevertheless not able to achieve a performance as high as that of Explicit Semantic Analysis (ESA, [17]). Our method, however, has clear advantages over ESA in that it provides a fully unsupervised approach that practically requires no tuning, and thus can be applied to arbitrary data and domains with virtually no changes. The original performance figures for ESA [17] have been criticized in fact for being based on a cut-off value used to prune the vectors being over-fitted to the LP50 data [4] – cf. also the much lower performance obtained by re-implementations like [4] and [22], among others. Thus, we take these figures to be promising in that our approach to document semantic similarity, while being based on a general document model with many potential applications – e.g. ranking related entities (cf. Section 3) – is nevertheless able to come close in performance to a highly specialized method like ESA, which has been tuned for this specific task and dataset.

**Error analysis.** In order to gain additional insights into the performance of our method, we performed an error analysis of its output. To this end, we focused on the manual analysis of documents deemed closest or most distant from the human judgments. When looking at specific document pairs, we found that our knowledge-rich approach is able to estimate well the similarity between documents with little or partial word overlap: connecting paths between DBpedia entities, in fact, were found to implicitly cover a wide range of topical associations, ranging from near-synonymity ("U.S. intelligence" and "CIA") all the way to metonymic relations ("White House" and "Bush administration"). However, since it relies only on DBpedia entities and their document mentions, our approach will perform badly in cases where i) the input documents contain few or no entities, or ii) they share the same entities, but describe different events. For instance, our method will give a very high similarity score to the following two sentences, although they describe completely different events: (a) Obama started his second term in the White House; (b) Obama will soon leave the White House. But while our approach could be extended to include relations between entities which are automatically extracted from text, cf. recent work on building event graphs from documents [20], our results seem also to suggest that in the case of text similarity we can often get away without a deep analysis of the documents' sentences, since entity overlap is a good proxy for topical affinity. This is highlighted by the following two sentences from the LP50 data, which, albeit very different, belong to documents which were deemed highly similar by the annotators:

- Nigerian President Olusegun Obasanjo said he will weep if a single mother sentenced to death by stoning for having a child out of wedlock is killed, but added he has faith the court system will overturn her sentence. [...]
- An Islamic high court in northern Nigeria rejected an appeal today by a single mother sentenced to be stoned to death for having sex out of wedlock. [...]

## 5. RELATED WORK

The recent years have seen a great deal of work on computing semantic similarity [49]. This is arguably because semantic similarity provides a valuable model of semantic compatibility that is widely applicable to a variety of complex tasks, including both pre-processing tasks like Word Sense Disambiguation [38] and coreference resolution [39], and high-end applications such as information retrieval [14] or multi-document summarization [33], to name a few.

Most of the previous work on semantic similarity has concentrated on computing pairwise similarity of words, although recent efforts concentrated on the broader task of text similarity [4], as also shown by community efforts such as the shared tasks on Semantic Textual Similarity [1]. Overall, the best results in these evaluation campaigns have been obtained by supervised models combining large feature sets [3, 45], although questions remain on whether this approach can be easily ported to domains for which no labeled data exists. In contrast, in this work we presented an unsupervised model that requires virtually no parameter tuning and exploits the implicit supervision provided by very large amounts of structured knowledge encoded in DBpedia.

This work is, to the best of our knowledge, the first to exploit a wide-coverage ontology (i.e., other than small-scale semantic lexicons like WordNet) within a general-purpose algorithm for computing semantic similarity based on a graph-based similarity measure. Our method effectively uses large amounts of structured knowledge and can be used in principle with other such resources like, e.g., YAGO [25], provided they contain explicit semantic relations. Seminal work on representing natural language as semantic networks focused on queries [7]. Recently, graph-based representations from DBpedia have been explored by [28] for labeling topics, as obtained from a topic model, rather than providing structured representations of arbitrary texts. In addition, they limit graph construction to a small set of manually selected DBpedia relations. The work closest to ours is that of [42], who use graph-based representations of snippets for Web search results clustering. Their method also builds a document-based semantic graph from Wikipedia concepts, as obtained from the output an entity disambiguator. However, similarly to [43], they do not exploit explicit semantic relations between entities (which we show to be beneficial for both entity ranking and semantic similarity).

Previous work in computing semantic distances on linked data relied on disambiguated input [37], a requirement which is very hard to satisfy for most applications working with natural language text. In contrast, our approach relies on automatic entity linking techniques, which allow us to link entity mentions in text to well-defined entities within an ontology. From a general perspective, our work can be viewed as building upon seminal research work in IR that explored the use of controlled vocabularies [30], originally introduced for library systems. The proposed method can thus be seen as instance of an advanced Knowledge Organization System (KOS) [48, 13], since it relies at its core on a wide-coverage ontology to represent documents. However, as opposed to these approaches, we do not create a controlled vocabulary for a specific document collection, but instead reuse an existing, background ontology which contains general world knowledge. We use this knowledge source to represent the entities found documents, as opposed to using the documents' headings or metadata. The Jaccard similarity we report in Section 4.3 consists, in fact, of a baseline method that uses DBpedia as controlled vocabulary: we build upon this intuition and extend it by using the information encoded within the structure of the DBpedia network.

## 6. CONCLUSIONS

In this paper, we proposed a method for exploiting large amounts of machine-readable knowledge, i.e., entities and semantic relations, encoded within DBpedia, in order to provide a structured, i.e. graph-based, representation of natural language texts. Our results on entity ranking and document semantic similarity indicate that, thanks to an effective weighting of the semantic relations found within the semantic network, as well as a robust concept matching technique, we are able to achieve competitive performance on both these hard NLP tasks, while at the same time providing an unsupervised model which is practically parameter-free – namely, whose only tunable parameter can be fixed based on well-established findings from previous work.

This is the first proposal to exploit a Web-scale ontology to provide structured representations of document content, and computing semantic distances in a knowledge-rich fashion. We build thematically upon previous contributions which showed the beneficial effect of exploiting large amounts of knowledge for enhancing text comparison [46, 35] (*inter alia*). In our work, we take this line of research one step further by: (a) using a truly ontological resource for content modeling (as opposed, e.g., to semantic lexicons such as WordNet); (b) developing an information-theoretic measure to identify semantically specific, highly informative relations between entities in a large knowledge graph; (c) defining a new method, based on graph edit distance techniques, to quantify degrees of semantic similarity between documents: this views semantic similarity as a concept matching problem and uses the Hungarian method for solving the combinatorial optimization problem.

Our vision, ultimately, is to show how entity linking and disambiguation techniques can enable an open-domain structured representation of documents, and accordingly an even larger Web of Semantic Data, on which semantic technologies (e.g., search) can be enabled. Accordingly, we focused in this first initial step primarily on entities, since they are the bulk of wide-coverage knowledge resources like DBpedia. Clearly, extending this entity-centric model – for instance, by means of event-structured graphs [20] or RDF predicates [19] – is the next logical step. Besides, as future work we plan to develop methods to jointly perform entity disambiguation and compute semantic similarity. We are also interested in applying our techniques within domains other than newswire data, and investigating domain adaptation techniques for the graph construction phase. Our graphs naturally model fine-grained information about documents: accordingly, we will explore their application to complex, high-end task such as aspect-oriented IR, as well as fine-grained document classification and clustering for IR.

## Acknowledgments

We thank our colleagues Arnab Dutta, Christian Meilicke and Heiner Stuckenschmidt for their valuable comments on draft versions of this work.

## 7. REFERENCES

[1] E. Agirre, D. Cer, M. Diab, A. Gonzalez-Agirre, and W. Guo. *SEM 2013 shared task: Semantic textual similarity. In *Proc. of *SEM-2013*, pages 32–43, 2013.

[2] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the Web. In *Proc. of IJCAI-07*, pages 2670–2676, 2007.

[3] D. Bär, C. Biemann, I. Gurevych, and T. Zesch. UKP: Computing semantic textual similarity by combining multiple content similarity measures. In *Proc. of SemEval-2012*, pages 435–440, 2012.

[4] D. Bär, T. Zesch, and I. Gurevych. A reflective view on text similarity. In *Proc. of RANLP-11*, pages 515–520, 2011.

[5] C. Bizer, T. Heath, and T. Berners-Lee. Linked data – the story so far. *International Journal on Semantic Web and Information Systems*, 2012.

[6] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia – A crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009.

[7] J. Booth, B. D. Eugenio, I. F. Cruz, and O. Wolfson. Query sentences as semantic (sub) networks. In *Proc. of ICSC-09*, pages 89–94, 2009.

[8] E. Bruni, J. Uijlings, M. Baroni, and N. Sebe. Distributional semantics with eyes: Using image analysis to improve computational representations of word meaning. In *Proc. of MM'12*, pages 1219–1228, 2012.

[9] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *Proc. of AAAI-10*, pages 1306–1313, 2010.

[10] M. Cornolti, P. Ferragina, and M. Ciaramita. A framework for benchmarking entity-annotation systems. In *Proc. of WWW-13*, pages 249–260, 2013.

[11] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[12] G. Demartini, T. Iofciu, and A. P. de Vries. Overview of the INEX 2009 entity ranking track. In *INEX*, pages 254–264, 2009.

[13] K. Eckert. *Usage-driven Maintenance of Knowledge Organization Systems*. PhD thesis, Universität Mannheim, 2012.

[14] O. Egozi, S. Markovitch, and E. Gabrilovich. Concept-based information retrieval using Explicit Semantic Analysis. *ACM Transactions on Information Systems*, 29(2):8:1–8:34, 2011.

[15] P. Ferragina and U. Scaiella. Fast and accurate annotation of short texts with Wikipedia pages. *IEEE Software*, 29(1):70–75, 2012.

[16] D. A. Ferrucci, E. W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefer, and C. A. Welty. Building Watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79, 2010.

[17] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proc. of IJCAI-07*, pages 1606–1611, 2007.

[18] X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Pattern Analysis and Applications*, 13(1):113–129, 2010.

[19] D. Gerber and A.-C. N. Ngomo. Extracting multilingual natural-language patterns for RDF predicates. In *Proc. of EKAW-12*, pages 87–96, 2012.

[20] G. Glavaš and J. Šnajder. Recognizing identical events with graph kernels. In *Proc. of ACL-13*, pages 797–803, 2013.

[21] I. Gurevych, J. Eckle-Kohler, S. Hartmann, M. Matuschek, C. M. Meyer, and C. Wirth. UBY – a large-scale unified lexical-semantic resource based on LMF. In *Proc. of EACL-12*, pages 580–590, 2012.

[22] S. Hassan and R. Mihalcea. Semantic relatedness using salient semantic analysis. In *Proc. of AAAI-11*, pages 884–889, 2011.

[23] J. Hees, M. Khamis, R. Biedert, S. Abdennadher, and A. Dengel. Collecting links between entities ranked by human association strengths. In *Proc. of ESWC-13*, pages 517–531, 2013.

[24] J. Hoffart, S. Seufert, D. B. Nguyen, M. Theobald, and G. Weikum. KORE: Keyphrase overlap relatedness for entity disambiguation. In *Proc. of CIKM-12*, pages 545–554, 2012.

[25] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

[26] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust disambiguation of named entities in text. In *Proc. of EMNLP-11*, pages 782–792, 2011.

[27] E. Hovy, R. Navigli, and S. P. Ponzetto. Collaboratively built semi-structured content and Artificial Intelligence: The story so far. *Artificial Intelligence*, 194:2–27, 2013.

[28] I. Hulpus, C. Hayes, M. Karnstedt, and D. Greene. Unsupervised graph-based topic labelling using DBpedia. In *Proc. of WSDM-13*, pages 465–474, 2013.

[29] H. Ji and R. Grishman. Knowledge base population: Successful approaches and challenges. In *Proc. of ACL-11*, pages 1148–1158, 2011.

[30] F. W. Lancaster. *Vocabulary Control for Information Retrieval.* Information Resources Press, 1972.

[31] M. D. Lee, B. Pincombe, and M. Welsh. An empirical evaluation of models of text document similarity. In *Proc. of CogSci 2005*, pages 1254–1259, 2005.

[32] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. DBpedia Spotlight : Shedding light on the web of documents. In *Proc. of I-Semantics'11*, pages 1–8, 2011.

[33] V. Nastase. Topic-driven multi-document summarization with encyclopedic knowledge and activation spreading. In *Proc. of EMNLP-08*, pages 763–772, 2008.

[34] R. Navigli and A. Di Marco. Clustering and diversifying Web search results with graph-based Word Sense Induction. *Computational Linguistics*, 39(3):709–754, 2013.

[35] R. Navigli, S. Faralli, A. Soroa, O. L. de Lacalle, and E. Agirre. Two birds with one stone: learning semantic models for Text Categorization and Word Sense Disambiguation. In *Proc. of CIKM-11*, pages 2317–2320, 2011.

[36] R. Navigli and S. P. Ponzetto. BabelNet: the automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012.

[37] A. Passant. Measuring semantic distance on linking data and using it for resources recommendations. In *Proceedings of the AAAI Spring Symposium "Linked Data Meets Artificial Intelligence"*, 2010.

[38] S. Patwardhan, S. Banerjee, and T. Pedersen. Using measures of semantic relatedness for Word Sense Disambiguation. In *Proc. of CICLing-03*, pages 241–257, 2003.

[39] S. P. Ponzetto and M. Strube. Knowledge derived from Wikipedia for computing semantic relatedness. *Journal of Artificial Intelligence Research*, 30:181–212, 2007.

[40] S. P. Ponzetto and M. Strube. Taxonomy induction based on a collaboratively built knowledge repository. *Artificial Intelligence*, 175:1737–1756, 2011.

[41] K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image Vision and Computing*, 27(7):950–959, 2009.

[42] U. Scaiella, P. Ferragina, A. Marino, and M. Ciaramita. Topical clustering of search results. In *Proc. of WSDM-12*, pages 223–232, 2012.

[43] W. Shen, J. Wang, P. Luo, and M. Wang. LINDEN: linking named entities with knowledge base via semantic knowledge. In *Proc. of WWW-12*, pages 449–458, 2012.

[44] P. D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188, 2010.

[45] F. Šarić, G. Glavaš, M. Karan, J. Šnajder, and B. Dalbelo Bašić. TakeLab: Systems for measuring semantic text similarity. In *Proc. of SemEval-2012*, pages 441–448, 2012.

[46] P. Wang and C. Domeniconi. Building semantic kernels for text classification using Wikipedia. In *Proc. of KDD '08*, pages 713–721, 2008.

[47] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum. Natural language questions for the web of data. In *Proc. of EMNLP-CoNLL-12*, pages 379–390, 2012.

[48] M. L. Zeng. Knowledge Organization Systems (KOS). *Knowledge Organization*, 35(2-3):160–182, 2008.

[49] Z. Zhang, A. L. Gentile, and F. Ciravegna. Recent advances in methods of lexical semantic relatedness – a survey. *Natural Language Engineering*, 1(1):1–69, 2012.