ABSTRACT

In recent years, countless research papers have addressed the topics of knowledge graph creation, extension, or completion in order to create knowledge graphs that are larger, more correct, or more diverse. This research is typically motivated by the argumentation that using such enhanced knowledge graphs to solve downstream tasks will improve performance. Nonetheless, this is hardly ever evaluated. Instead, the predominant evaluation metrics - aiming at correctness and completeness - are undoubtedly valuable but fail to capture the complete picture, i.e., how useful the created or enhanced knowledge graph actually is. Further, the accessibility of such a knowledge graph is rarely considered (e.g., whether it contains expressive labels, descriptions, and sufficient context information to link textual mentions to the entities of the knowledge graph). To better judge how well knowledge graphs perform on actual tasks, we present KGrEaT – a framework to estimate the quality of knowledge graphs via actual downstream tasks like classification, clustering, or recommendation. Instead of comparing different methods of processing knowledge graphs with respect to a single task, the purpose of KGrEaT is to compare various knowledge graphs as such by evaluating them on a fixed task setup. The framework takes a knowledge graph as input, automatically maps it to the datasets to be evaluated on, and computes performance metrics for the defined tasks. It is built in a modular way to be easily extendable with additional tasks and datasets.

1 INTRODUCTION

1.1 Motivation

Knowledge graphs (KGs) have emerged as a powerful tool for organizing and representing structured knowledge in a machine-readable format. Starting with Google’s announcement of the Google Knowledge Graph in 2012, research articles have extensively explored the creation [3, 18, 24], extension [10, 14], refinement [22], and completion [1] of KGs, with the aim of producing larger, more accurate, and more diverse graphs. These efforts are driven by the belief that leveraging enhanced KGs can lead to improved performance in downstream tasks. However, comparative evaluations of different KGs w.r.t. their utility for such tasks are rarely conducted.

In the literature, the vast majority of studies concerned with the evaluation of KGs have focused on intrinsic metrics that are working exclusively with the triples of a graph. Several works introduce quality metrics like accuracy, consistency, or trustworthiness and propose ways to determine them quantitatively [4, 8, 16, 33, 35]. Färber et al. [7] and Heist et al. [11] compare KGs with respect to size, complexity, coverage, and overlap. Additionally, they provide guidelines on which KG to select for a given problem.

Another line of work computes extrinsic task-based metrics to evaluate KG embedding approaches. They use a fixed input KG with a fixed evaluation setup while varying only the embedding approach. Frameworks like GEval [23] or kgbench [5] use data mining tasks like classification or regression for the evaluation, others, like Ali et al. [2] evaluate primarily on link prediction tasks.

1.2 Contributions

To address the evaluation gap of extrinsic metrics for KGs, we propose a framework called KGrEaT (Knowledge Graph Evaluation...
KGrEaT aims to provide a comprehensive assessment of KGs by evaluating them on multiple kinds of tasks like classification, regression, or recommendation. The evaluation results (e.g., the accuracy of a classification model trained with the KG as background knowledge) serve as extrinsic task-based quality metrics for the KG. By defining a fixed evaluation setup in the framework and applying it to multiple KGs, it is possible to isolate the effect of every single KG and compare how useful they are for solving different kinds of tasks. KGrEaT is built in a modular way to be open for extensions from the community like additional tasks or datasets.

Overall, the contributions of this paper are as follows:

- With KGrEaT, we present a framework to judge the utility of KGs using extrinsic task-based metrics (Section 2).
- In our experiments, we demonstrate the capabilities of the framework in an evaluation and comparison of several well-known cross-domain KGs (Section 3).

## 2 FRAMEWORK

### 2.1 Purpose and Limitations

KGrEaT is a framework built to evaluate the performance impact of KGs on multiple downstream tasks. To that end, the framework implements various algorithms to solve tasks like classification, regression, or recommendation of entities. The impact of a given KG is measured by using its information as background knowledge for solving the tasks. To compare the performance of different KGs on downstream tasks, we use a fixed experimental setup with the KG as the only variable. The performance indicators may be used to make an informed decision when picking a KG for a given task. Further, they can be used to compare the performance of different versions of a single KG (e.g., during construction or during its life cycle).

The implemented algorithms are not necessarily state-of-the-art because the primary objective is not to measure how well a task can be solved with a given KG in absolute numbers, but rather in comparison to other KGs or different versions of the same KG. Hence, the absolute numbers of the results only have limited expressive power. However, the framework tries to reduce the bias in the results by averaging over multiple preprocessing methods, datasets, and algorithms.

KGrEaT maps the entities of the KG automatically to the entities of the dataset using a set of configurable mappers. Undoubtedly, the quality of this mapping influences the results generated by the framework. But as the mapping procedure is applied similarly to all evaluated KGs, the mapping quality is mainly influenced by the accessibility of the graph (i.e., whether it provides sufficient context information like labels or descriptions for its entities). To reduce the influence of the mapping strategy on the overall results, the framework provides a way to run experiments with multiple mapping approaches (and possibly average over them).

### 2.2 Design

The framework is designed in a modular way (c.f. Figure 1), making it easy to add additional preprocessing steps, mappers, or tasks. Every step of a stage is implemented as an isolated docker container with its own environment so that additions can be made without any constraints on the programming language. Another advantage of the container-based architecture is the easy distribution of containers via a container hub, eliminating the need for users to build the framework on their own machines.

The manager is responsible for making necessary preparations (e.g., downloading the input data or gathering entities to be mapped), executing the stages (fetching and running containers of the steps), and visualizing the results (e.g., comparing KG performance on various aggregation levels). The Preprocessing and Mapping stages can be executed in parallel, and the results are then used to execute the Task stage. The whole process can be steered via a command line interface (CLI).

The only input to the evaluation process is the KG in the form of RDF files as well as a configuration. The latter defines how the stages should be run (i.e., which steps to execute in which order). Further, every step can be configured in depth to supply relevant hyper-parameters. For example, one can configure how the KG should be mapped to the datasets (e.g., via matching labels) and define an acceptable similarity value for a match.

In the following, we provide details of the three main stages that are executed when running an evaluation of a KG.

### 2.3 Preprocessing Stage

The Preprocessing stage creates all pre-computable artifacts that are needed in the subsequent Task stage (e.g., intermediate representations or statistics of the KG). So far, this stage comprises the computation of embeddings (TransE [6], TransR [17], DistMult [34], REascal [20], CompleX [32] via the DGL-KE framework [37], and RDF2vec [28] via the jRDF2vec framework [25]). Further, it supports the generation of indices for Approximate Nearest Neighbor (ANN) search (via the hnswlib library [19]).

### 2.4 Mapping Stage

In the Mapping stage, the entities of the KG are automatically mapped to the entities in the datasets. So far, a Same-As mapper and a Label mapper are implemented. The former uses the same-as links of a KG to map its entities to those of the datasets. A dataset may provide URIs for an entity (e.g., from well-known KGs like DBpedia or Wikidata), but it has to provide at least one label. This label is used by the Label mapper to find a corresponding entity in the KG. It uses the RapidFuzz library to estimate the similarity of labels via token-based edit distance. Mappers are composable, i.e., they can be executed in sequence. For example, entities are first mapped via same-as links where available, and the remaining entities are mapped via label similarity.

### 2.5 Task Stage

In the Task stage, the task types are executed for all combinations of datasets and algorithms. Table 1 gives an overview of all possible constellations. In total, KGrEaT contains 26 tasks (i.e., combinations of task types and datasets) that are run with one or more algorithms.
Figure 1: An overview of the KGrEaT framework.

Table 1: Implemented tasks together with their algorithms, datasets, and evaluation metrics.

<table>
<thead>
<tr>
<th>Task Type</th>
<th>Datasets</th>
<th>Algorithms</th>
<th>Evaluation Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>AAUP, Cities, Forbes, MillionSongDataset, MetacriticAlbums, MetacriticMovies, ComicCharacters</td>
<td>Naive Bayes, KNN, SVM</td>
<td>Accuracy ↑</td>
</tr>
<tr>
<td>Regression</td>
<td>AAUP, Cities, Forbes, MetacriticAlbums, MetacriticMovies</td>
<td>Linear Regression, KNN, Decision Tree</td>
<td>RMSE ↓</td>
</tr>
<tr>
<td>Clustering</td>
<td>Cities2000AndCountries, CitiesAndCountries, Teams, CitiesMoviesAlbumsCompaniesUni, ComicCharacters</td>
<td>DBSCAN, KMeans, Agglomerative Clustering</td>
<td>ARI ↑, NMI ↑, Accuracy ↑</td>
</tr>
<tr>
<td>Document Similarity</td>
<td>LP50</td>
<td>Cosine Similarity</td>
<td>Spearman ↑, Pearson ↑</td>
</tr>
<tr>
<td>Entity Relatedness</td>
<td>KORE</td>
<td>Cosine Similarity</td>
<td>Kendall’s Tau ↑</td>
</tr>
<tr>
<td>Semantic Analogies</td>
<td>AllCapitalCountryEntities, CapitalCountryEntities, CityStateEntities, CurrencyEntities</td>
<td>Cosine Similarity</td>
<td>Accuracy ↑</td>
</tr>
<tr>
<td>Recommendation</td>
<td>MovieLens, LastFm, LibraryThing</td>
<td>Item-Similarity recommender</td>
<td>F1 ↑</td>
</tr>
</tbody>
</table>

Additionally, the algorithms are executed with multiple hyperparameter settings. How the individual tasks use the KG information is dependent on the task and the implemented algorithm. Generally, the tasks Classification, Regression, and Clustering use embeddings of the KG’s entities as features of the models, and the remaining tasks use the distance between the entity embeddings to find related entities.

Several datasets are taken from Ristoski et al. [27] and from the GEval framework [23]. The Recommendation datasets MovieLens [9], LastFm, and LibraryThing [36] are preprocessed as recommended by Di Noia et al. [21] with the exception of using all entities instead of only those for which a mapping to DBpedia exists. For detailed statistics of all datasets, please refer to the respective publications and the information in the framework.

Every task type comes with suitable evaluation metrics that are computed for every constellation. As some KGs might not contain matches for all entities in the dataset and it would not be fair to compute metrics only over known entities (and discard unknown entities) or only over all entities, the framework reports metrics for both scenarios. Finally, the results can be aggregated over various levels (e.g., over embeddings, algorithms, and datasets) to produce metrics with a reduced bias.

3 EXPERIMENTS

To show the capabilities of KGrEaT, we conduct experiments over multiple large cross-domain KGs and analyze how well they perform on the implemented downstream tasks. We first give an overview of the evaluated KGs, then define the experimental setup, and finally discuss the results.

3.1 Experimental Setup

3.1.1 Knowledge Graphs. We use the following KGs in our experiments:

1. DBpedia [3]: Dumps from 2016-10 and 2022-09
2. YAGO [18]: Version 3
3. Wikidata [24]: Dump from 2023-06-07
4. CaliGraph [12, 13]: Version 3.1.1
5. DBkWik [15]: Version DBkWik++

Using multiple versions allows us to compare not only between different KGs but also between different versions (here: with respect to time) of the same KG.

Combined with DBpedia to also include the well-known entities of Wikipedia.
3.1.2 Mapping. We first map the KGs with the Same-As mapper where applicable. Then we apply two variants of the Label mapper: One with a similarity threshold of 1.0 for high-precision matches and one with a threshold of 0.7 for high recall. For the former, we compute metrics for known entities (Precision Known - PK) and for all entities (Precision All - PA); for the latter, being recall-oriented, we report the metrics only for all entities (Recall All - RA).

3.1.3 Embeddings. To reduce the influence of the different embedding approaches on the overall results, all experiments are executed with four embedding types (TransE, DistMult, ComplEx, and RDF2Vec). For Wikidata, we could not compute all these embeddings due to the amount of computational resources necessary. Instead, we use pre-computed TransE embeddings\(^1\) with a comparable training configuration.

3.1.4 Hardware. All experiments are executed on NVIDIA RTX 2080 Ti graphic cards and Intel Xeon E5 processors (2.6GHz). On average, a full evaluation of a single KG takes roughly 30 hours with 20 hours of embedding computation, 4 hours of mapping, and 6 hours of task execution.

3.2 Results and Discussion

Table 2 shows the final results of our evaluation for the three scenarios PK, PA, and RA. The results are averaged after aggregating over all embeddings, datasets, and algorithms. The complete results of the experiments are publicly available.\(^2\)

For Classification, DBpedia2016 shows the best results in the precision setting, while CaliGraph and YAGO achieve the best results in the recall setting. For Regression, both DBpedia versions and Wikidata perform well in the precision setup, while again YAGO and CaliGraph achieve the best results in the recall setting. The Clustering task is solved best by DBpedia2016, YAGO, and DkbWik. For Document Similarity, version 2022 of DBpedia is the clear winner. For the Entity Relatedness task, using DBpedia2022, Wikidata, or DkbWik as background knowledge produces the best results. Recommendation is solved best using DBpedia or Wikidata, Semantic Analogies is also solved best by DBpedia.

In general, DBpedia dominates the results to a large extent which may be explained by the fact that some of the datasets used in the framework have been derived from the 2015 version of DBpedia. This might also explain that there is no clear advantage of the 2022 version of DBpedia over the older 2016 version. However, both versions of DBpedia perform strongly on the Recommendation task which has no direct relation to DBpedia or even Wikipedia.

Our assumption that the KGs with more entities (YAGO, Wikidata, CaliGraph, and DkbWik) will have an advantage, especially in the Recommendation tasks, did only partially prove to be true. However, they have shown strong performances, especially in recall-oriented settings. A reason for this unsteady performance may lie in the increased complexity of training expressive embeddings for large KGs. In the future, we want to explore this further by running evaluations not only with multiple types of embeddings but also with multiple embedding configurations (e.g., number of trained epochs).

Another interesting direction to explore is whether combining two KGs (e.g., by concatenating their entity vectors) yields improved results [31].

4 CONCLUSION AND OUTLOOK

We presented KGrEaT, a framework for evaluating the performance of KGs on multiple downstream tasks. In our experiments, we found that, depending on the task, the performance of the KGs varies enormously. To judge the quality of a KG in its completeness, extrinsic evaluation metrics provided by KGrEaT can serve as a valuable addition to the established intrinsic evaluation criteria.

In the future, we want to improve the framework in various ways, e.g., by providing more embedding methods such as RDF2Vec [29] as well as more tasks like KG Question Answering [30].

Further, we plan to include a more comprehensive mapper that uses all information of an entity (such as comments and relations to other entities). To that end, we transform the entities of the datasets into a small KG which is then mapped to the entities of the KG under evaluation. In such a case, systems participating in the Ontology Alignment Evaluation Initiative (OAEI) [26] may prove useful.

To open the framework for users unfamiliar with programming and docker, we will introduce a graphical user interface, allowing them to analyze KGs in a faster and more intuitive way.
REFERENCES


