KnowlyBERT - Hybrid Query Answering over Language Models and Knowledge Graphs

Jan-Christoph Kalo(✉), Leandra Fichtel, Philipp Ehler, and Wolf-Tilo Balke

Institut Für Informationssysteme, Technische Universität Braunschweig, Mühlenpfordtstraße 23, 38106 Braunschweig, Germany
{kalo,balke}@ifis.cs.tu-bs.de, {l.fichtel,p.ehler}@tu-bs.de

Abstract. Providing a plethora of entity-centric information, Knowledge Graphs have become a vital building block for a variety of intelligent applications. Indeed, modern knowledge graphs like Wikidata already capture several billions of RDF triples, yet they still lack a good coverage for most relations. On the other hand, recent developments in NLP research show that neural language models can easily be queried for relational knowledge without requiring massive amounts of training data. In this work, we leverage this idea by creating a hybrid query answering system on top of knowledge graphs in combination with the masked language model BERT to complete query results. We thus incorporate valuable structural and semantic information from knowledge graphs with textual knowledge from language models to achieve high precision query results. Standard techniques for dealing with incomplete knowledge graphs are either (1) relation extraction which requires massive amounts of training data or (2) knowledge graph embeddings which have problems to succeed beyond simple baseline datasets. Our hybrid system KnowlyBERT requires only small amounts of training data, while outperforming state-of-the-art techniques by boosting their precision by over 30% in our large Wikidata experiment.

Keywords: Query answering · Language models · Knowledge graphs

1 Introduction

Large Knowledge Graphs (KG) like Wikidata [17], DBpedia [2], YAGO [15] and the Google Knowledge Graph [5] have become an essential component in data-intensive applications, like Web search, information retrieval or for adding an additional value to machine learning techniques. Most of these knowledge graphs contain entity-centric knowledge that is either manually curated like in Wikidata, extracted from structured sources like tables for DBpedia or also extracted from natural language text by relation extraction techniques [5].

J. Z. Pan et al. (Eds.): ISWC 2020, LNCS 12506, pp. 294–310, 2020.
https://doi.org/10.1007/978-3-030-62419-4_17

Still, modern knowledge graphs like Wikidata lack important information about entities which drastically hampers its application. As an example, only 36% of all persons in the current Wikidata version have a birthplace. To overcome problems with the incompleteness, several ways to complete knowledge graphs are investigated: (1) Knowledge graph completion techniques, as for example relational learning techniques [10], are employed for learning statistical regularities in knowledge graph data to infer new facts. However, current benchmarks, only comprising several thousand entities, show that existing techniques are far from being able to deliver reliable results [1]. (2) On the other hand, relation extraction method use existing triples as training data for NLP machine learning techniques to extract similar facts from textual data automatically. These techniques need large amounts of training data and also only achieve low quality results [13].

Very recently Petroni et al. have proposed a third idea for dealing with the incompleteness of structured knowledge graphs: utilizing masked language models as a knowledge graph [13]. Masked language models are a technique that lead to a quantum leap in almost all natural language processing tasks. It is a machine learning technique, that is able to complete sentences, based on massive amounts of text that it was trained on. The language model is even able to complete sentences correctly, if the information was not present in the training data, since it is able to infer new knowledge. Therefore, Petroni et al. came up with the idea of extracting relational knowledge (i. e; triples) directly from the model. As an example, we could figure out the birthplace of Albert Einstein by a sentence completion task as follows: "*Albert Einstein is born in ...*", would be completed by the language model with the word *Ulm*. Hence, we could infer the valid triple (*Albert Einstein*, `born in`, *Ulm*) to directly answer basic SPARQL queries on the language model. Some follow up works [3,12], have shown that we easily extract relational knowledge from such language model, with good quality for a variety of semantic and syntactic relations. However, this idea still shows several problems: In contrast to knowledge graphs, language models work on words and not on IRIs for entities as it is common in knowledge graphs. This leaves the open question on how to map the language model result to the correct knowledge graph entity. Existing works, are only considering entities whose label consists of a single word. Multi-word entities cannot be found. So far, only first experiments on how triples are extracted from a language model have been shown.

Here, instead of just extracting knowledge from the language model, we overcome existing problems and present - KnowlyBERT - the first read-to-use hybrid query answering system for knowledge graphs incorporating masked language models on the fly at query time. Our system is able to answer entity-centric SPARQL queries on incomplete knowledge graphs, using state-of-the-art language models to complete the results. Furthermore, we incorporate existing semantic information from the knowledge graph in multiple ways to improve the accuracy of the hybrid system: We developed a typing system, filtering language model results based on knowledge graphs fine-grained class hierarchies.

As an additional filtering step, our system uses existing answers to further filter out incorrect answer candidates. We also use the knowledge graphs information to automatically find good natural language sentence for the language model prediction task [3].

We present the first ready-to-use query answering system for Wikidata combined with the language model BERT [4]. In a large real-world evaluation on Wikidata, we compare our system for various queries on an incomplete knowledge graph against a state-of-the-art relation extraction system ([14]) and the knowledge base completion system HoLE [11].

The main contributions of this paper can be summarized as follows:

- We develop a system integrating the advantages of knowledge graphs and large-scale masked language models to answer entity-centric SPARQL queries efficiently.
- We perform several large-scale experiments with around 6500 queries of KnowlyBERT on the real-world knowledge graph Wikidata against two state-of-the-art baselines.
- Our implementation and all our experimental data is openly available for easy reproducibility of our work[1].

2 Related Work

Knowledge Graph Completion is used to learn statistical regularities in the data to predict new triples connecting existing entities in incomplete knowledge graphs. Thus, similar to our system, such techniques may be used to complete query results, by finding missing triples in a first step and use these triples additionally, to existing ones to answer queries.

Basically, two approaches for knowledge graph completion are common. Rule-based approaches, such as AMIE+ [7] are used to learn closed Horn rules from knowledge graphs. These rules infer new triples with high precision. However, rule induction algorithms have performance problems when it comes to large knowledge graphs with hundreds of millions of triples such as Wikidata.

To overcome the performance problem, knowledge embedding-based techniques have been proposed. These techniques have in common that they learn high-dimensional vector representations of entities and relationships and use these to predict new triples [10]. In practice, knowledge embeddings are hardly used yet because their result quality for real-world knowledge graphs is poor and it is not desirable to introduce low quality triples into a high-quality knowledge graph [1]. Furthermore, state-of-the-art benchmarks rely on a well-chosen subset of the knowledge graph Freebase, only comprising 15,000 entities. The prediction quality in these benchmarks is acceptable, but mainly because of the small and well-chosen datasets.

Overall, existing techniques for knowledge graph completion cannot work properly with large-scale knowledge graphs and achieve high quality predictions

[1] https://github.com/JanKalo/KnowlyBERT.

at the same time. In contrast to both techniques, our approach makes use of pre-trained language models without the need of massive computing resources and still can predict new triples with high quality or high recall.

To provide a comparison, we also evaluate the state-of-the-art KG embedding technique HoLE [11] against our system in the evaluation section.

Relation Extraction is about extracting triples from natural language text using automatic machine learning techniques. State-of-the-art techniques are based on so called *distant supervision*. Existing knowledge graph triples are used to generate training data for a classification algorithm that decides for a given sentence whether it contains a triple or not. Systems achieve a precision between 50% and 90% on small Wikidata corpora [14,16]. In contrast to our work, relation extraction systems need large amounts of training data to train a classifier for each relation independently. Furthermore, they cannot be directly used in an on-the fly querying answering system. Relation extraction has to be performed upfront on large text collections to cover all possible queries, because due to its runtime it cannot be used for on-the-fly query answering. To provide a comparison to relation extraction, in our evaluation, we compare to the open source system provided by Sorokin et al. [14].

Masked Language Models have recently shown great results in a plethora of different NLP tasks. Petroni et al. have shown that knowledge graph facts can be directly extracted from pre-trained language models, instead performing a complex relation extraction process [13]. In their work, they have manually built sentence templates for several relationships from knowledge graphs. These templates are then used to complete a sentence and predict a word to complete a triple, as demonstrated in the introduction. This work is only able to predict entity labels consisting of a single word, excluding almost all persons from being a query answer. Furthermore, it only provides words as answers, but not KG IRIs. An entity linking step is not performed yet. They evaluate on several relations from Wikidata, but also on a variety of other relation datasets. They achieve a high accuracy of 32% on the T-Rex datasets [6], comprising 41 different Wikidata relations. We evaluate our system on the same relations from T-Rex, but a larger dataset also comprising multi-word entities.

In an extension, it was shown that the quality of these predictions is highly dependent on the input sentence. Therefore Bouraoui et al. have proposed to automatically generate templates for relationships that achieve high quality results [3]. They show that indeed they can double the accuracy of triple inference for some relations. Still, this approach is also restricted, since it cannot predict entities consisting of more than a single word.

Another work in this direction circumvents the restriction to single-word entities, by defining a new fine-tuning task on masked language models [18]. They directly include entity knowledge from the knowledge graph to train masked language models. However, in their work they restrict to a small amount of most popular entities from Wikidata only. Covering a large amount of entities, including rare entities would dramatically increase the computational effort for

training. Thus, this approach is not suitable for general query answering in knowledge graphs.

In contrast, our present work builds upon existing systems and further refines their ideas to create a query answering system for incomplete knowledge graphs without any restrictions on the type of entities or relations.

3 Preliminaries

KnowlyBERT is a query answering system for RDF-based knowledge graphs. RDF information is expressed in the form of subject, predicate, object triples $(s, \mathtt{p}, o) \in E \times P \times E \cup L$, whereas s represents an entity from the real-world E, p represents a relation from the set of all relations P that can be mapped to real-world relations and o represents an entity from E or a literal value from L. Entities and predicates in RDF are represented by Internationalized Resource Identifiers (IRIs). Furthermore, each entity from E and each relation from P has a natural language text label. For readability reasons in this work, we use these labels instead of IRIs. As an example, the triple (*Albert Einstein*, `bornIn`, *Ulm*) states that the entity Albert Einstein is in a `born in` relation with the entity *Ulm*. Entities may have multiple types from the set of classes $C \subseteq E$. A type relation can also be formulated in a triple as follows: (*Albert Einstein*, `type`, *Scientist*) The set of all triples in a knowledge graphs defined as $KG \subseteq E \times P \times E \cup L$.

Queries against a KG can be performed by the query language SPARQL. In this work, we restrict to the most prominent part of SPARQL: basic graph pattern (BGP) queries. The simplest basic graph pattern query consists of a single triple pattern with variables that are indicated by a leading ?. As an example, a BGP query $Q = (\textit{Albert Einstein}, \mathtt{bornIn}, ?x)$ is asking for the `birthplace` of *Albert Einstein*. To answer the query, we need to match this triple against the knowledge graph, whereas a variable can be matched to an arbitrary entity. In the example case, $?x$ could be matched to the entity *Ulm*. In the present work, we restrict to entity-centric SPARQL queries. They have one triple pattern with a single variable either in subject or object position.

A language model is a statistical model providing the probability of an upcoming word, given a sequence of words. Usually, it is used to complete natural language sentences, by learning the parameters of the model from large text corpora. In this work, we work with the masked language model BERT, a neural network-based language model, which is based on the transformer model [4]. This model is created in a pre-training step, where the model learns to complete arbitrary sentences and predict next sentences in a self-supervised fashion using large input text corpora, e.g. Wikipedia. Thus it, for example, learns how to complete the sentence: "*The birthplace of Albert Einstein is....*" with the word *Ulm*. Therefore, masked language models, as BERT, may be used to answer simple basic graph pattern queries, as demonstrated by Petroni et al. [13]. As an example, we could translate the query $Q = (\textit{AlbertEinstein}, \mathtt{bornIn}, ?x)$ to the sentence "*Albert Einstein is born in ...*", which would be completed with

the word *Ulm*. In most masked language models, the word to be predicted is represented by a so-called mask token "[MASK]". The input to the language model is a sentence including a mask token: "*The birthplace of Albert Einstein is* [MASK]". The model's output is a list of words from its vocabulary, accompanied with a prediction probability for each of them.

Finding appropriate translations from triples to sentences, so that the language model is able to answer the query is an open problem that has gained recent interest [3,9,12]. We further investigate this problem in the present work by comparing different schemes for sentence generation. As an additional problem, masked language models, as they are available today, are only able to predict words, but not entities. So, to answer a query, we need to map natural language words to an entity from our knowledge graph in E. Because of that, some language model extensions have been developed to include entity knowledge. As described in our related work section, this leads to problems, which is why we stick to masked language models and develop an entity disambiguation technique on top. Existing works for extracting relational knowledge from language models have focused on entities having labels consisting of a single word only. We will extend the technique to multi-word entities (entities whose label consists of more than a single word). More details of these step will be described in the next section.

4 KnowlyBERT - Query Answering with Language Models and Knowledge Graphs

In this section, we describe KnowlyBERT, a hybrid query answering system using a knowledge graph and the masked language model BERT to complete queries over an incomplete knowledge graph. We start with an overview of the system and shortly describe all components, while we go into the details of the different components in the following subsections.

The overview is sketched in Fig. 1. As an input for KnowlyBERT, a user can pose an entity-centric SPARQL query to our system. First, the language model is queried (a). Then we pose the query to the incomplete knowledge graph and get the existing results (b). The SPARQL query is translated into multiple natural language sentences that are completed by the language model in the Relation Template Generation step. As a result, the language model returns multiple lists of words together with a confidence value for each word (c). These lists are then combined into a single list (d) and filtered using our semantic filtering step based on knowledge graph type information (e). Furthermore, we perform a thresholding, cutting of irrelevant results (f). As a final step, the results of the language model and the knowledge graph are combined (g) and returned to the user.

4.1 Relation Template Generation

As a first step to query a language model for relational knowledge, a SPARQL query needs to be translated into a natural language sentence with [MASK]

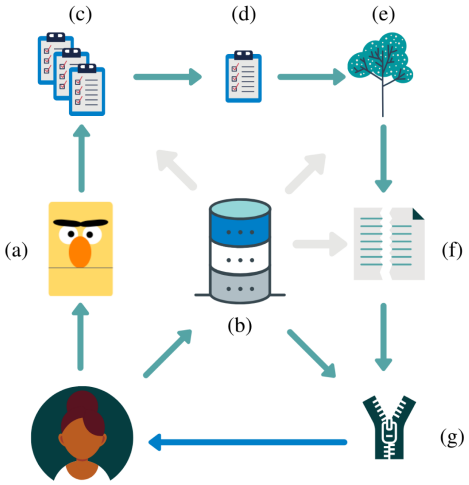

Fig. 1. An overview of the query answering system KnowlyBERT. A user query is distributed to BERT using the generated templates from Sect. 4.1 (a) and the KG (b). BERT outputs several result lists (c) with information from the KG. The results are integrated and filtered as described in Sect. 4.2 (d). A semantic type filter (Sect. 4.3) is applied (e) and later we employ thresholding methods to cut off incorrect results (f), which is described in Sect. 4.4. Finally, results from the LM and KG are integrated (g) and returned to the user.

tokens. Bouraoui et al. showed in [3] that automatically generated sentences outperform manually build templates as presented by Petroni et al. [13]. Therefore, in this work, we adapt the idea and automatically extract and rate sentence candidates for each relation of a knowledge graph to generate relation sentence templates in a pre-processing step. Such a template may have the format: "*[S] is born in [O]*", for the `birthplace` relation whereas [S] is replaced by the subject entity of a query or the [O] by the object. Generating sentence templates is not performed at query time, but is a pre-processing step.

To demonstrate this procedure, we continue with our artificial running example from Sect. 3: $(\textit{Albert Einstein}, \texttt{bornIn}, ?x)$. For the relation `birthplace`, we use all subject, object pairs (s, o) such that $(s, \texttt{bornIn}, o) \in KG$. An example is the subject, object pair $(GottfriedLeibniz, Leipzig)$. We use pre-annotated Wikipedia abstracts (T-REx), where entity recognition and disambiguation as well as relation linking have been performed already [6]. As sentence candidates, we extract all sentences containing exactly one (s, o) pair for the `bornIn` relation. For example, the sentence *"Gottfried Leibniz was born in the city of Leipzig, Germany".* for the respective entity pair. As template sentences, we use sentences with at most 15 words and exactly 2 tagged entities. Longer sentences often are too specific or consist of multiple subordinate clauses from different contexts. Hence, choosing different parameters here, may decrease the performance of the predictions. In contrast to the idea in [3], our input sentences have also been

processed by basic co-reference resolution in sentences, replacing pronouns by the respective entity names as provided by the T-REx dataset [6].

As an additional new technique, we perform a string similarity check between the different sentences for the `bornIn` relation and only pick sentences that are different to each other to end up with a diverse set of sentence templates for each relation. We use a basic sequence pattern matching method finding the longest sub-sequence between every two template sentences. When the similarity of such a pair is above 0.8, it is put into the same sentence cluster. After performing similarity checks among all sentences, we obtain a set of sentence template clusters comprising similar sentences. From each cluster only a single sentence template is picked as a representative.

In a final step, we rate sentence template, using the language model and valid (s, o) pairs from the KG similar to Bouraoui et al. [3]. We rate the extracted `birthplace` sentence in the following way. The sentence *"Gottfried Leibniz was born in the city of Leipzig".* is instantiated by subjects and objects, so that we can check the language model's predictions and compare to the correct `bornIn` pairs from the KG. For example, we create the sentence *"Gottfried Leibniz was born in the city of [MASK]"* using the existing pair (*Gottfried Leibniz, Leipzig*). We have a look at the top predictions of the language model and check, whether they have an overlap with existing objects of the `birthplace` relation in the KG. The size overlap of these object predictions with the KG objects and size of the overlap of the respective subject predictions are summed up and used as a weight of the sentence template. In the end, we take the top five sentence templates for our predictions.

Additional Context Paragraphs. An additional boost in the prediction quality of language models is achieved by providing additional context information to the query sentences [12]. Petroni et al. have shown that instead of only giving a query template sentence as the input to the language model, providing additional sentences about the entity of interest increases the precision of predictions by around 30%. Following their description, for each entity in a query, we have extracted the first five sentences from the respective Wikipedia abstract and added them to the generate templates using a [SEP] token of BERT. We provide context with the first sentences of Einstein's Wikipedia article as follows:

> *Albert Einstein was born in the city of* [MASK].[SEP]
> *Albert Einstein was a German-born theoretical physicist who developed the theory of relativity, one of the two pillars of modern physics (alongside quantum mechanics).*

In contrast to existing works, we combine automatic template generation and context paragraph retrieval, which in combination boost the result quality. We use five template sentences for each query, each annotated with a five sentence long context paragraph.

Template generation for relations and also context paragraph retrieval can be performed in a pre-processing step. The information for all relations and entities in the KG is indexed properly for fast access at query time.

4.2 Querying the Language Model and Combining the Results

For the Einstein query, we now use multiple sentence templates together with the respective context paragraph to get possible answers from the language model. Since possible answer entity labels might consist of more than a single word, we have to pose queries with a single [MASK] token for returning possible single-word entities, but also queries with multiple [MASK] tokens. Changing our example query to (?x, `bornIn`, Ulm), would require us to find (among others) Albert Einstein, an entity with a label of two words, as a correct answer. To predict such entities, we provide two or more [MASK] tokens in the query sentence as follows: "[MASK] [MASK] *was born in the city of Ulm*" for a template returning two tokens and "[MASK] [MASK] [MASK] *was born in the city of Ulm*" if we want to find answer entities with three tokens. This leads to independent result list for each [MASK], so that we have to check for valid word combinations. We join all possible word combinations from the result lists and check whether a valid entity label from the knowledge graph is created. This is a very important step, since it enables us to filter out a large proportion of predicted words which cannot be mapped to any entity. Valid word combinations are weighted by the average of the words output probabilities. The output is a list of single and multi-token entity labels with the correctness probabilities assigned by the masked language model. We present an artificial result set as an example for the query containing one correct entity in the top position and two locations below:

1. *Ulm - 0.95*
2. *Princeton, New Jersey - 0.45*
3. *Munich - 0.22*

Note that in this work we restrict the system to a maximum of three [MASK] tokens. This number may be increased with small increases in the query answering time.

Aggregating Results from Multiple Templates. Different sentence templates for a single query lead to independent result lists with different probability values for each result entity. In our case, we obtain 5 entity lists. To combine these lists into a single list, we stick to the idea from [3]. The lists are first simply merged. If an entity occurs in multiple lists the maximum probability is chosen. Furthermore, the maximum probability and the minimum probability for each entity occurring in multiple lists are compared. If their difference exceeds a threshold of 0.6, the entity is not taken into the final result list. 0.6 has been chosen as a good compromise between precision- or recall-oriented behavior. This excludes entities, where the language model shows unstable predictions, leading to a higher overall precision. Increasing this threshold increases the overall precision of the results.

4.3 Semantic Type Filtering

Most knowledge graphs provide a very detailed type hierarchy for its entities which we employ for further filtering the language model results. For every relation in the knowledge graph, we create frequency distributions over the classes

of the subject and objects entities. A class of an entity is determined by using the `instance of` relation (P31) of Wikidata, but we also consider the first level of super classes (P279) of these classes, so that the type filter is not too specific. Using the created frequency distributions, we only define the most frequently occurring classes as valid types for a relation. These can be called the expected types of a relation. Based on the expected types, we are able to filter out answer candidates that do not fit these types, employing a semantic type filter. This way, we are more restrictive, possibly filtering out some correct entities, but increase the systems precision.

Entity Disambiguation. After the semantic type filtering step, we still could end up with multiple possible answer entities, having the same entity label. Such a homonym could for example be another entity with the name *Ulm*. In Germany as an example, one large city, but also several smaller villages called *Ulm* are known. For such rare cases, we need to perform an additional entity disambiguation step. As a first simple pre-processing step, we exclude extremely rare entities using a popularity filter. Concretely, entities are excluded when they never occur as an object entity in the whole knowledge graph. If multiple homonyms exist, the most popular entity is returned as an answer. Further filtering steps using knowledge graph embeddings are possible here. However, our evaluation has shown no benefits in precision without large losses in recall here.

4.4 Thresholding

As a last step before returning the result list, we perform a thresholding procedure to guarantee that only high-quality results are returned to the user. We perform two different thresholding mechanisms. The first threshold is dynamically chosen for each query by a statistical outlier analysis among the prediction values. If after several top prediction values (e.g. 0.95, 0.45, 0.22), the next prediction value is significantly lower (e.g. 0.45), we pick a threshold in the gap between 0.95 and 0.45. If no correct answer is returned by the language model, the dynamic threshold methods do not work. Therefore, we pick an additional static threshold that is valid for all queries. This threshold is learned automatically by averaging the probabilities of known results that are already in the incomplete knowledge graph and that are also in the language model's result list. Using our example result list from above, we end up with the following list only comprising the correct answer Ulm:

1. *Ulm - 0.95*

Finally, we join the result lists of the incomplete knowledge graph with the result list of our language model-based pipeline and eliminate duplicates.

5 Evaluation

In this section, we describe the evaluation of KnowlyBERT on the large real-world knowledge graph Wikidata and the language model BERT. We evaluate

precision and recall for 41 different relations similar to other language model-based systems [3,12,13] and compare against a state-of-the-art relation extraction technique using distant supervision [14] and a technique for knowledge graph completion which uses high dimensional embeddings [11]. In detail, we provide an overview of the performance on different relations and provide an extensive discussion on the drawbacks and advantages of language model-based techniques for on-the-fly query answering in contrast to existing techniques which are particularly trained for inducing new triples in incomplete knowledge graphs.

5.1 Experimental Setup

Baselines. KnowlyBERT performs query answering on incomplete knowledge graphs, which may be seen as an on-the-fly knowledge graph completion method. Since no directly comparable baselines are available, we compare to standard knowledge graph completion techniques that work in an offline setting. Here, inferring new triples using external knowledge by relation extraction from text and triple induction by structural methods purely on the knowledge graph are the most popular methods being used today.

Therefore as a first baseline, we use a recent distant supervised relation extraction system from [14] with available pre-trained models for Wikipedia triple extraction. This baseline has already been used by Petroni et al. [13] to compare to their language model-based approach. We have used their pre-trained Wikipedia model for extracting triples from natural language text and performed relation extraction from T-Rex [6]. T-Rex links Wikidata entities and triples to Wikipedia abstracts. These linked entities in text are used as an input for the relation extraction framework, to extract triples from sentences.

As a second baseline, we have compared to another state-of-the-art technique for coping with incomplete knowledge graphs [10]. Knowledge graph embeddings are latent machine learning models for knowledge graph completion. High dimensional vector representations of entities and relations are learned from an existing knowledge graph. Arithmetic operations between these vector representations enable giving a correctness probability to every possible subject, predicate, object-combination. Hence, it is also possible to find most likely substitutions for subject-predicate-pairs or predicate-object pairs. In our case, we use HoLE as a baseline, which has shown good results in benchmark datasets, and also is scalable to the size of our large Wikidata sample [11]. Due to the size of Wikidata, we trained HoLE using 50 dimensions for 200 epochs. Since HoLE itself only provides a top-k list of newly inferred triples ordered by their prediction probability, we only took the predictions with the best possible prediction value of HoLE. This may also include several predictions showing the same prediction value.

Dataset. Our experiments are performed on the Wikidata Truthy dump from February 6th, 2020. We evaluate only on triples where subject and object are entities having an `rdf:label` relation. For simplicity reasons, we also restrict to labels consisting of at most three words. We restrict to the 41 different relations

that are used in the LAMA probe [13]. But we use different queries, since they were restricted to entities consisting of single word labels only.

We have sampled queries for each of these 41 relations by randomly choosing triples from the Wikidata. We remove the subject, creating an entity-centric SPARQL query, asking for a subject entity (?x, p, o), or removing the object respectively to ask for the object (s, p, ?x). We name the query type asking for subjects, *subject queries* and the others *object queries*. Hence, we created 100 subject and 100 object queries for each relation, if possible. For some relations, we could only generate fewer queries. Overall, this leads to 6649 queries.

For all queries we assume that the current Wikidata version as the ideal knowledge graph. The incomplete knowledge graph is simulated by leaving out existing triples by performing a *leave k out* evaluation, deleting at least 1 and at most 100 answers from the answer set of each query. To be comparable to the relation extraction baseline which extracts triples from text, we have restricted the deleted triples to triples that actually occur in the text corpus we use. This gives an advantage to this baseline system since it ensures that it is possible to achieve 100% recall which is not necessarily valid for our system. The ideal knowledge graph has 54,056,746 triples and the incomplete knowledge graph has 125,213 fewer triples deleted for the 6,649 queries. Thus, the incomplete knowledge graph comprises 53,931,533 triples.

Evaluation Metrics. We have evaluated every query separately by querying the language model and removing the answer triples that already were in the incomplete KG. For the remaining additional results, we computed precision and recall values. The reported results are average precision and recall values over all queries that returned additional results.

Implementation Details. Our system KnowlyBERT is implemented in Python 3 and is openly available on Github[2]. We also make scripts for reproducing these results available. Our system is based on the masked language model BERT from Google [4]. We use the large and cased model pre-trained by Google comprising 340 m parameters. Since our system is built on the LAMA framework by Petroni et al, we are able to include arbitrary language models[3]. For the relation extraction baseline, we use the original implementation also available on Github[4]. The knowledge graph embedding HoLE is implemented in OpenKE [8].

5.2 Experimental Results

An overview of precision and recall of KnowlyBERT and the two baseline systems is presented in Table 1. First, we have a look at the total precision and recall values depicted in the last row. KnowlyBERT outperforms the two other approaches with regard to precision by more than 30% by achieving an average precision of 47.5%. In contrast to the relation extraction baseline (RE), we

[2] https://github.com/JanKalo/KnowlyBERT.
[3] https://github.com/facebookresearch/LAMA.
[4] https://github.com/UKPLab/emnlp2017-relation-extraction.

Table 1. Precision (Prec) and Recall (Rec) from KnowlyBERT against two baseline systems in percent. Relation extraction (RE) and the knowledge graph embedding technique HoLE (KE) on 41 relations. We evaluate different query parameters.

Evaluation	Parameter	Statistics		RE		KE		KnowlyBERT	
		#Queries	#Rel	Prec	Rec	Prec	Rec	Prec	Rec
Cardinality	1-1	400	2	5.5	5.5	<0.1	**20.2**	**16.9**	3.0
	1-n	3756	23	18.8	**17.4**	<0.1	11.5	**55.0**	13.7
	n-m	2493	16	16.4	19.8	<0.1	**22.6**	**36.0**	5.9
Query type	$(s, p, ?x)$	4029	41	37.5	17.3	<0.1	**20.5**	**51.0**	16.5
	$(?x, p, o)$	2620	41	6.9	**17.9**	<0.1	9.5	**10.5**	0.3
Words	single	2474	41	39.6	13.9	<0.1	21.1	**59.6**	**25.9**
	multi	4175	41	**13.0**	**19.7**	<0.1	13.2	11.4	0.8
#Results	1	3497	41	40.5	13.2	<0.1	15.8	**51.3**	**17.4**
	2–10	1367	39	18.7	**20.5**	<0.1	20.4	**37.0**	4.9
	11–100	796	37	7.4	**30.7**	0.2	24.7	**15.8**	0.1
	>100	989	37	**5.7**	**18.2**	<0.1	4.8	<0.1	<0.1
Total		6649	41	17.5	**17.6**	<0.1	16.2	**47.5**	10.1

improve the precision drastically, however the recall of our approach is slightly lower with 10.1% in comparison to 17.6% of the RE baseline.

HoLE (KE) is showing good results with regard to the recall, but its precision is extremely low at around 0.03%. This very low precision, but high recall value is due to a high number of false positives all having top prediction values. The result for a knowledge graph embedding technique confirms recent research results that it is not ready for completion tasks in real-world knowledge graphs [1]. We present the results here anyways for completeness reasons, but will not discuss them in detail. Our main focus in this evaluation will compare the RE baseline with KnowlyBERT.

In the first rows, we present the results ordered by the cardinality of the relations in the query. We have analyzed two 1-1 relations[5], 23 1-n relations and 16 n-m relations. KnowlyBERT show its best results for 1-n relations with a precision of 55.0% and recall of 13.7%. Similarly, the two baselines show there best precision here.

We also present an evaluation of subject vs. object-based queries. Here, we observe something particularly interesting. KnowlyBERT achieves an extremely high precision for (s, p, ?x) queries asking for the object, but low precision and recall for queries asking for the subject of a triple. Also, the RE baseline shows a much smaller precision here, but at least shows good recall values.

[5] We follow the categorization of Petroni et al. [13]. Note that some queries for 1-1 relations have more than a single result.

Table 2. Precision (Prec) and Recall (Rec) of KnowlyBERT and the baseline systems for a variety of relations from Wikidata in percent.

Relation	Label	Statistics	RE		KE		KnowlyBERT	
		#Queries	Prec	Rec	Prec	Rec	Prec	Rec
P17	`country`	145	16.6	16.7	<0.1	21.6	**97.4**	**51.0**
P19	`birthplace`	191	21.8	**19.4**	<0.1	13.7	**73.3**	11.5
P31	`instance of`	152	**11.9**	15.0	<0.1	**17.3**	<0.1	<0.1
P36	`capital`	200	5.5	11.1	<0.1	**23.0**	**15.4**	3.0
P101	`field of work`	174	11.0	9.3	<0.1	**12.1**	**45.1**	7.8
P103	`native language`	117	<0.1	<0.1	<0.1	31.5	**100**	**74.3**
P108	`employer`	173	17.1	3.2	<0.1	**17.3**	**100**	0.6
P159	`headquarter`	190	19.6	**26.8**	<0.1	9.5	**56.8**	13.2
P279	`subclass of`	197	6.8	**28.8**	<0.1	13.5	16.7	<0.1
P1303	`instrument`	128	**35.0**	**43.4**	<0.1	15.8	<0.1	<0.1
P1412	`language spoken`	124	6.4	2.5	<0.1	**21.9**	**45.8**	17.7

The next part of our evaluation presents on how well the different approaches deal with multi word entities. We analyze whether the respective results of queries who only return single word entities against queries which correct answers comprise also multi-word entities. Here, we observe that KnowlyBERT works best for single-word entities, as does the RE baseline. Multi-word entities are often much harder to find using a language model-based approach. One reason is that queries asking for persons are often multi-word queries. Answering such person queries, is extremely difficult, since the set of possibly correct answers is often huge. Details will be evaluated in the next category in Table 1.

The next evaluation clusters queries by the number of results they have in the ideal KG. Here, we see that queries with few results generally show much better results. The precision and recall of KnowlyBERT for queries with a single result is over 50% with a recall of over 17% achieving the best results. Queries with large result sets are only answered with a low result quality. If they have more than 100 answers, we hardly find any correct answer, resulting in a poor precision. The RE baseline also has worse results for queries with many results, but at least returns some results.

In Table 2, we present the results for some particularly good working and badly working relations. We see that for some relations we achieve a precision of over 90% and a recall also above 70%. We see that many of the well working relations are about locations or languages. On the other hand, we also have several relations with an extremely low recall near to 0%. Particularly bad were the `instance of` and `subclass of` relations. Which implies that type information is hardly represented in the language model. But also the `instrument` relation shows extremely bad results. In contrast, the RE baseline shows its best results here.

5.3 Discussion

The evaluation of KnowlyBERT in comparison to other techniques for coping with incomplete knowledge graphs has shown us, that none of the existing techniques is ready to deal with all problems that come with missing information. While the knowledge graph embedding-based technique has shown poor results in the real-world scenario as already shown in recent research on the evaluation of such techniques [1], the state-of-the-art relation extraction technique has shown a consistently moderate result quality with a precision and recall around 17%.

In contrast, a language model-based approach shows a much higher precision with a small loss in recall. We have seen that the language model BERT has very different quality depending on the relation used in the queries. In some cases, we achieve almost perfect results with a precision of over 90% and high recall values, whereas for other relations we cannot find any correct results at all. Particularly geographic relations show good results, outperforming the baselines by far. Queries with single-word entities are also showing good quality. However, multi-word entities are very difficult to predict. Multi-word queries strongly correlate to queries with large result sets and subject-queries. One possible problem is that subject queries and multi-word queries often ask for long-tail entities. For these, the language model is rarely able to provide correct answers. All of these problems are reflected by our lower recall in contrast to the baselines. Particularly the relation extraction baseline still achieves an acceptable recall for these difficult query types.

Note that queries with large result sets are substantially more difficult to solve. Due to the fact that we do not count predicted result entities that already are in the incomplete KG, we add another difficulty. Even though a technique finds correct results, its precision for such queries might be 0%.

6 Conclusion

In this work, we have presented a hybrid query answering system for knowledge graphs using language models to cope with the incompleteness of real-world knowledge graphs. We have seen a plethora of different techniques to find missing triples in a knowledge graph completion task in a pre-processing step. Such techniques would introduce a lot of new and often incorrect triples into the knowledge graph, since they are not producing high quality results. Knowledge graph embedding techniques only show high precision in standard benchmark datasets, but fail in large real-world knowledge graphs [1]. On the other hand, NLP methods, that extract triples from natural language text in a binary classification task, require massive amounts of training data and a high quality entity linking step up front. The quality of relation extraction is under 20%, which would introduce massive amounts of incorrect data.

In contrast, we have presented a precision-oriented method that does not extract triples in a pre-processing step to be inserted into the knowledge graph, but an on-the-fly query answering system. This way, we do not contaminate the

high quality of a knowledge graph and still can help to provide complete results, if necessary. KnowlyBERT has shown that a language model is a promising way to reduce the gap between incomplete knowledge graphs and complete result sets when used in combination with the KG itself. The KG can help to filter many incorrect results from the language model and helps us to also return multi-word entities, enabling us to be used as a full-fledged query answering system. As a drawback, we have seen that some relations could not be answered at all by KnowlyBERT, since the language model did not return correct results.

So far, KnowlyBERT is restricted to basic entity-centric queries. Existing question answering datasets could be used to learn natural language query templates to feature more complex queries with multiple triples. Furthermore, we plan to further investigate the boundaries of language models in representing relational knowledge to further characterize its benefits in knowledge graph tasks. Additionally, the suitability of general purpose language models in more specific domains should be investigated. In specific domains, choosing a domain-specific language model (e.g. BioBERT, SciBERT) and different method for retrieving contextual paragraphs would be interesting fields of research.

References

1. Akrami, F., Saeef, M., Zhang, Q., Hu, W., Li, C.: Realistic re-evaluation of knowledge graph completion methods: an experimental study. In: Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, Portland, OR, USA, 14–19 June 2020, March 2020
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
3. Bouraoui, Z., Camacho-Collados, J., Schockaert, S.: Inducing relational knowledge from BERT. In: Proceedings of the Thirty-Fourth Conference on Artificial Intelligence, AAAI 2020 (2020)
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 4171–4186, Jun 2019
5. Dong, X., et al.: Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In: Proceedings of the 20th International Conference on Knowledge Discovery and Data Mining, SIGKDD 2014, pp. 601–610 (2014)
6. Elsahar, H., et al.: T-REx: a large scale alignment of natural language with knowledge base triples. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan. European Language Resources Association (ELRA), May 2018
7. Galárraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast rule mining in ontological knowledge bases with AMIE+. VLDB J. **24**(6), 707–730 (2015)
8. Han, X., et al.: OpenKE: an open toolkit for knowledge embedding. In: EMNLP (2018)
9. Jiang, Z., Xu, F.F., Araki, J., Neubig, G.: How can we know what language models know? Trans. Assoc. Comput. Linguist. (TACL) **8**, 423–438 (2020)

10. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. In: Proceedings of the IEEE, vol. 104, no. 1, pp. 11–33, January 2016
11. Nickel, M., Rosasco, L., Poggio, T.: Holographic embeddings of knowledge graphs. In: Proceedings of the 30 AAAI Conference on Artificial Intelligence, AAAI 2016, pp. 1955–1961. AAAI Press (2016)
12. Petroni, F., et al.: How context affects language models' factual predictions. In: Automatic Knowledge Base Construction (AKBC) (2020)
13. Petroni, F., et al.: Language models as knowledge bases? In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 2463–2473, November 2019
14. Sorokin, D., Gurevych, I.: Context-aware representations for knowledge base relation extraction. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 1784–1789, September 2017
15. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, p. 697 (2007)
16. Trisedya, B.D., Weikum, G., Qi, J., Zhang, R.: Neural relation extraction for knowledge base enrichment. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 229–240, July 2019
17. Vrandečić, D.: Wikidata: a new platform for collaborative data collection. In: Proceedings of the 21st International Conference Companion on World Wide Web, WWW 2012 Companion, p. 1063 (2012)
18. Xiong, W., Du, J., Wang, W.Y., Stoyanov, V.: Pretrained encyclopedia: weakly supervised knowledge-pretrained language model. In: International Conference on Learning Representations (ICLR) (2020)