IntKB: A Verifiable Interactive Framework for Knowledge Base Completion

Bernhard Kratzwald[♢] **Guo Kunpeng**[♠♡] **Stefan Feuerriegel**[♢] **Dennis Diefenbach**[♠♡]
[♢] Chair of Management Information Systems, ETH Zurich
[♠] CNRS, Laboratoire Hubert Curien UMR 5516, University of Lyon
[♡] The QA Company SAS, France
{bkratzwald, sfeuerriegel}@ethz.ch
{kunpeng.guo, dennis.diefenbach}@the-qa-company.com

Abstract

Knowledge bases (KBs) are essential for many downstream NLP tasks, yet their prime shortcoming is that they are often incomplete. State-of-the-art frameworks for KB completion often lack sufficient accuracy to work fully automated without human supervision. As a remedy, we propose *IntKB*: a novel interactive framework for KB completion from text based on a question answering pipeline. Our framework is tailored to the specific needs of a human-in-the-loop paradigm: (i) We generate facts that are aligned with text snippets and are thus immediately verifiable by humans. (ii) Our system is designed such that it continuously learns during the KB completion task and, therefore, significantly improves its performance upon initial zero- and few-shot relations over time. (iii) We only trigger human interactions when there is enough information for a correct prediction. Therefore, we train our system with negative examples and a fold-option if there is no answer. Our framework yields a favorable performance: it achieves a hit@1 ratio of 29.7% for initially unseen relations, upon which it gradually improves to 46.2%.

1 Introduction

Knowledge bases (KBs) present databases that store information about entities and the relations among them. Prominent examples are, for instance, Wikidata (Vrandečić and Krötzsch, 2014), Freebase (Bollacker et al., 2008), NELL (Carlson et al., 2010), DBpedia (Auer et al., 2007), or Yago (Suchanek et al., 2008). Owed to their structured information, KBs are widely used in various downstream tasks of NLP such as, e. g., entity linking (Mendes et al., 2011), query expansion (Dalton et al., 2014), co-reference resolution (Rahman and Ng, 2011), and question answering (Diefenbach et al., 2019). Yet the performance of downstream NLP tasks is often impeded due to the issue that KBs are incomplete. For example, for newspapers, Wikidata lists the place of publication for only 37% of them, while the owner is stored for only a staggering 9%. Hence, there is a need to develop scalable strategies for KB completion.

Prior work on KB completion can be divided into two paradigms. On the one hand, KB completion via link prediction. Here the structural information of an existing, yet incomplete KB is leveraged in order to predict new facts (Lao et al., 2011; Riedel et al., 2013; Neelakantan et al., 2015). On the other hand, KB completion can be built upon a text corpus, so that new facts are extracted directly from the narrative materials (Dong et al., 2014; West et al., 2014). Approaches belonging to the latter paradigm are scarce, as they usually rely upon some way of aggregation information from multiple, different corpora (where credible additions to a KB are signaled by cross-corpus appearances). In contrast, we develop a framework for KB completion over a single corpus, where such verification is obtained in an interactive manner.

Knowledge base completion is often designed as an automated task where predictions are directly integrated into an existing but incomplete KB. This can be problematic for various reasons: (i) The performance of state-of-the-art systems is not of sufficient accuracy, so that it allows for an automatic integration of new facts (Akrami et al., 2020). Hence, manual verification is still necessary in practice. (ii) Predictions in KB completion often involve zero- or few-shot settings (e. g., relations for which there

Proceedings of the 28th International Conference on Computational Linguistics, pages 5591–5603
Barcelona, Spain (Online), December 8-13, 2020

is none or only or few examples available during training), this further diminishes their performance (Wang et al., 2019). (iii) State-of-the-art systems do not generate facts that are verifiable by humans because their predictions are mainly based on structural similarity within graphs. Thus, it is impossible for humans to manually detect false-positive predictions. This problem becomes especially critical in domains as, e.g., medicine, where KBs serve as the basis for high-stake decisions. To this end, KB completion should allow for simple human verification. Yet, to the best of our knowledge, research on verifiable KB completion is lacking.

In this paper, we propose *IntKB*: a novel interactive framework for KB completion based on question answering (QA). Unlike previous work, we design our framework to fulfill a human-in-the-loop paradigm. In *IntKB*, new candidate facts are presented to human annotators, who can then approve or reject the candidates before they are integrated into the KB. Based on this setting, we overcome several challenges that are inherent to prior work: First, candidate predictions need to be human verifiable. It is not sufficient to show human annotators simply a new KB entry, as it would require a substantial effort to verify its correctness. Therefore, we align predictions with text snippets that make them directly verifiable. Second, in order to tackle the low performance in cold-start (e.g., zero-shot) settings, we design our framework in such a way that it gradually improves over time from past human interactions. Here human-approved predictions are fed back into our training framework. Third, *IntKB* is designed for an interactive use with the aim of requiring only little manual effort. For this, it only triggers human interactions when there is sufficient evidence (i. e., the system can decide to fold otherwise).

Our framework is realized on the basis of a question answering (QA) pipeline. The prime benefit of this is that it operates on only a single text corpus (rather than multiple) when generating new facts with high confidence. In addition, we develop a training framework based on knowledge transfer via fact alignment and continuous updates. It allows us to make predictions in cold-start settings. It also achieves a favorable precision when predicting long-tail relations that are rare.

We evaluate our system in an extensive set of experiments. Overall, we annotated around 3000 new facts from human interactions. We demonstrate that our framework continuously learns from those interactions: Over time, the hit@1 ratio on initially unseen zero-shot relations increases from 29.7% to over 46.2% . A live demo is available from `https://wikidatacomplete.org`, the source code is available from `https://github.com/bernhard2202/intkb`.

2 Related Work

Recent research on KB completion can be divided into two main paradigms: (i) KB completion via link prediction based on the graph structure of an existing yet incomplete KB and (ii) KB completion using free text from the web or a corpus. Both are detailed in the following.

KB completion via link prediction: Here the idea is that new relations for a KB can be inferred by reasoning over the existing structure. For example, if a newspaper is published in Paris, then one can infer with high confidence that it is written in French. Formally, this can be achieved in multiple ways. One option are path-ranking algorithms, where paths between entities are used to predict new relations (Lao et al., 2011; Gardner and Mitchell, 2015; Zupanc and Davis, 2018). Another line of works uses matrix factorization for this purpose (Riedel et al., 2013; He et al., 2015; Lacroix et al., 2018). Even other works draw upon embeddings tailored to KBs. Here the the learned vector representation of entities and relations is used to infer new facts (Neelakantan et al., 2015; Bordes et al., 2013; Garcia-Duran and Niepert, 2017). More recently KB completion has been formulated as a reinforcement learning task (Das et al., 2017; Xiong et al., 2017) or attempted to be solved by variational inference (Chen et al., 2018). Another line of research combines textual description of entities and relations in order to create more accurate embeddings, that are then combined with methods for link prediction (Toutanova et al., 2015; Zhong et al., 2015; Xie et al., 2016).

Recent research on link prediction has increasingly focused on zero- and one-shot predictions (Xiong et al., 2018; Chen et al., 2019; Wang et al., 2019), where, for a specific relation type, only one or even no training samples are available. Such settings are demanded in practice because many relations that are in common KBs have only few associated triples. This especially holds true for so-called infrequent

long-tail relations, which are harder to predict in a one-shot setting than other prevalent relations (Wang et al., 2019).

To this end, the performance of state-of-the-art approaches lacks sufficient accuracy to deploy them in a fully automated setting (Akrami et al., 2020).

KB completion from free text: KB completion can also be achieved by extracting new relations from external text corpora (i. e., the web or other free text). These works, e.g., NELL (Carlson et al., 2010) or the knowledge vault (Dong et al., 2014) extract new facts from text sources using rule-based or traditional machine learning classifiers (Weikum and Theobald, 2010). Other works such as West et al. (2014) are tailored to web-based settings. To this end, they complete KB facts by aggregating and then scoring text-snippets that were provided by a search engine. These approaches make confident predictions by aggregating information from multiple data sources and, therefore, struggle with making predictions from a single corpus.

KB completion with question answering: Carlson et al. (2010) have previously formulated KB completion over free text as a QA task. In addition, recent advances in machine learning have lead to huge progress in machine reading comprehension (Rajpurkar et al., 2016) and QA (Chen et al., 2017). Owed to this, QA has experienced an increasing popularity as a format in related NLP tasks; see the overview in (Gardner et al., 2019). One example is the work by Das et al. (2019), who use machine reading comprehension to build dynamic knowledge bases from procedural text. Closest to our approach is the seminal work by Levy et al. (2017), who use machine reading comprehension for relation extraction in zero-shot settings.

Verifiability in KB completion: Few works have studied KB completion over free text with a focus on explainability or verifiability. An exception are rule-based systems (Weikum and Theobald, 2010), yet, these systems are mostly dependent on handwritten rules and thus inflexible. There has been some work on explainable KB completion, but this usually comes at the cost of a accuracy-explainability tradeoff (Stadelmaier and Padó, 2019). In contrast to that, state-of-the-art systems for KB completion lack verifiability.

Learning from user feedback: Learning from user feedback is an established technique to continue increasing the performance of systems after their deployment. Here researchers usually follow a human-in-the-loop methodology where explicit or implicit user feedback is fed back to the system. Learning from user feedback has been applied throughout various areas in natural language processing as information retrieval (Bendersky et al., 2017), semantic parsing (Yao et al., 2020), machine translation (Saluja et al., 2012), or question answering (Kratzwald and Feuerriegel, 2019).

3 Preliminaries

Knowledge Base: A knowledge base is given by a tuple $\langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V}$ is a set of entities and $\mathcal{E}$ is a set of relations between them (i. e., so-called "facts"). Relations themselves are triplets $\langle h, r_j, t \rangle \in \mathcal{E}$, where $h \in \mathcal{V}$ is the head entity, $t \in \mathcal{V}$ is the tail entity, and r_j is the relation type. Relationships in our setting are specified by one (or multiple) names $[\omega_0^{(j)}, \ldots]$ denoting their semantic meaning. For instance, the relationship referring to a company owner could be given by r_5 : `["is owned by","belongs to",...]`. Similarly, all entities $a \in \mathcal{V}$ are specified by one (or multiple) names $[\psi_0^{(a)}, \ldots]$. An example for the New York Times would be `["NYT", "NY Times", "New York Times",...]`.

Text Corpus: In addition, we are given a text corpus $\mathcal{D} = [d_1, \ldots d_n]$, where d_i refers to the i-th document. Documents are associated with entities and, to this end, we denote the document associated with entity $a \in \mathcal{V}$ by $d^{(a)}$. In our experiments, this corresponds to the Wikipedia page that describes an entity.

4 *IntKB*: A Verifiable Framework for Interactive KB Completion

This section describes our general framework (see Fig. 1) for interactive KB completion over free text. The input to our system is a query consisting of a head-entity and relation-type. We then wish to annotate

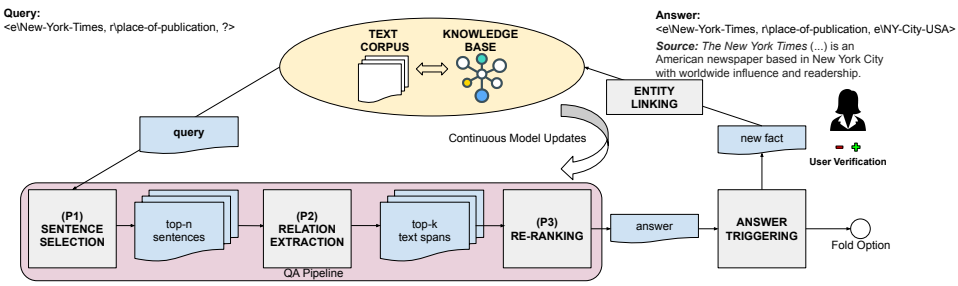

Figure 1: Overview of our QA pipeline consisting of modules (P1)–(P3) for knowledge base completion. The answer to a given query should be presented and, in addition, a textual source where the answer was extracted from. This makes our KB completion verifiable by humans.

the ground-truth tail entity (see Sec. 4.1). For this, we customize a QA pipeline. The QA pipeline comprises of three modules for sentence selection, relation extraction, and answer re-ranking (see Sec. 4.2). Finally, we use a module for answer-triggering that decides on whether to invoke a human interaction or end the prediction process by folding (see Sec. 4.3). In the case that a human interaction is triggered, we show the new candidate fact to human annotators and they can either approve or decline it. For approved predictions, we link the textual mention back to the actual entity (see Sec. 4.4) and add the new fact to the KB. Finally, we use the feedback from the human annotators to continuously update our framework (see Sec. 5).

4.1 Task Definition

The input to our KB completion framework is a given by a small incomplete knowledge base $\langle \mathcal{V}, \mathcal{E} \rangle$ and a text corpus $\mathcal{D}$. Based on them, our objective is to complete the KB by adding new facts to $\mathcal{E}$. Analogous to prior literature (Wang et al., 2019), we focus on the task of predicting the tail entity t: given a query consisting of a head entity h and relationship type r, we fill-in the missing tail entity in $\langle h, r_j, \_ \rangle$.

4.2 QA Pipeline for KB Completion

(P1) Sentence selection: The module for sentence selection receives a query in the form of $\langle h, r_j, \_ \rangle$. It then returns a ranked list of candidate sentences (i. e., a list of sentences that is likely to contain the textual answer). As in prior research (Hewlett et al., 2016), we constrain the search space to the document describing the head entity $d^{(h)}$ and split it into sentences, i. e., $d^{(h)} = [s_1, s_2, \ldots]$. Next, we create a tf-idf vector representation for each sentence. Thereby, we treat $[s_1, \ldots]$ as a closed corpus, that is, we are not using the full corpus $\mathcal{D}$ to calculate idf values.

For the query vectors, we use the semantic names of the relationship type $r_j = [\omega_0^{(j)}, \ldots]$ and transform every $\omega_i^{(j)}$ to its corresponding tf-idf vector. We then compute pairwise similarity scores for every sentence and every query vector. Finally, we return the top-n sentences with the highest score $[s^{(1)}, \ldots, s^{(n)}]$ (here: $n = 20$) to module (P2) for relation extraction.

(P2) Relation extraction: The module for relation extraction receives the top-n sentences $[s^{(1)}, \ldots, s^{(n)}]$ from the previous module in addition to the initial query $\langle h, r_j, \_ \rangle$. Each sentence is translated into a search query (as detailed later) and then fed into BERT-QA (Devlin et al., 2019) in order to return a text span as a candidate answer.

Following Levy et al. (2017), we treat relation extraction as a special case of reading comprehension, that is, we "ask" for the unknown tail entity. Different from earlier work, we do not require manually-annotated question templates in natural language (e. g., *"Where has [X] been published?"*). Instead, we construct multiple, noisy keyword queries from the semantic names of the relationship type $\omega_i^{(j)}$ (e. g., *"Place of publication?"*. Therefore, our KB completion operates in a data-scarce setting where only the information present in our KB is used and no additional human supervision is required.

In more detail, we pair every sentence $s^{(k)}$ with every relationship name $\omega_i^{(j)}$ and construct a BERT query by concatenating the name, a question mark, a separation token, and the sentence itself. Formally,

this is specified by

$$\{\omega_i^{(j)} + \texttt{"?"} + \texttt{"[SEP]"} + s_k\},$$

for all $i \in \{1, \dots |r_j|\}$ and $k \in \{1, \dots, n\}$, where $|r_j|$ denotes the number of semantic names of relation. This results in $n \times |r_j|$ queries. For each, BERT predicts both the start and the end point of the answer (a_s, a_e) using a softmax layer over words in the sentence. For every query, we extract the top-scoring span. Altogether, the module then results in up to $n \times |r_j|$ candidate answers for our initial query.

(P3) Answer re-ranking: The answer re-ranking module receives all $n \times |r_j|$ candidate answers for our initial query and re-ranks them in order to achieve a higher performance.

For re-ranking, we customize the approach in (Kratzwald et al., 2019) to our pipeline. That is we extract features for each candidate answer from both of the two aforementioned modules. Before re-ranking, we aggregate all candidate answers which have a matching text span. For instance, if the answer span "New York Times" is contained five times within our predictions, we then aggregate these predictions into a single candidate answer. Thereby, we additionally collect aggregated features (sum, average, mean, count, etc.). Finally, we take the top-10 aggregated candidate answers and predict their final score by using the answer re-ranking network proposed in (Kratzwald et al., 2019). We then return the top-scoring candidate answer as our final prediction for $\langle h, r_j, \_ \rangle$.

4.3 Answer Triggering

In *IntKB*, human feedback is requested only sparsely. That is, we integrate a component that can either fold and, thereby, decide not to pass a query on to a human annotator or trigger a human interaction. This is necessary for KB completion from free text, where the answer is not guaranteed to exist in the text corpus. In that case, the query becomes unanswerable and we do not wish to trigger costly user interactions. Our answer triggering module decides whether to trigger a human interaction or whether to fold on a query and thus not to pass it on to human annotators.

The above fold mechanism is implemented as follows. We train our framework with synthetic examples that were made unanswerable (Devlin et al., 2019; Rajpurkar et al., 2018) by removing any mention of the ground-truth answer from the text corpus. In this case, we set the correct start and end point of the answer in the relation extraction module to the `"[SEP]"` token (see Sec. 4.2 for more details). At prediction time, we check if the probability of the `"[SEP]"` token being the answer exceeds a threshold ϵ. If yes, we invoke the fold option and, otherwise, we trigger human interactions asking for feedback on the proposed fact.

4.4 Entity Linking

In order to perform the full task of KB completion, the text spans returned by the QA pipeline must be matched back to an entity $t \in \mathcal{V}$ of the KB. Here we experiment with three different strategies as follows. (i) Rank-based: The first strategy is to find all entities which have a label corresponding to the text span. These are then ordered based on the page rank of the KB (Diefenbach and Thalhammer, 2018). Finally, we return the top-ranked entity. (ii) Range-based: This strategy is an adaption of the previous one. The idea is that KBs generally contain meta-information about the range of the tail-entity in a relation. For example, the range of the relation "`award received`" is "`award`", which indicates that the tail-entity should belong to the type "`award`". Based on this information, we then exclude all entities that do not match the type of the range in the list of possible entities of the rank-based strategy. (iii) ML-based: We use a learning-to-rank model is to re-rank all candidate entities. The re-ranker uses as features all properties (i. e., instance-of) that are associated to a candidate entity in the KB. The idea is that the re-ranker learns to up-rank entities associated with particular concepts. Similarly to the second strategy, for the relation "`award received`", the rank model should learn to up-rank entities that are of type "`award`". As a result, concepts can be taken into account more flexibly and one does not rely on the completeness and exactness of the range properties in the KB.

5 Continuous Learning for KB Completion

This section explains how we learn the components inside our *IntKB* framework during (1) initialization as part of a cold-start and (2) continuous improvements from user feedback.

5.1 Cold-Start: Learning KB Completion from Text

In the following, we suggest a three step procedure to initialize our framework: First, we perform fact-alignment. This aligns facts from an initial KB with sentences from the text corpus using distant supervision. Second, we generate negative training samples for facts that are not present in our data (to learn the fold mechanism). Third, we perform a knowledge transfer in which the negative samples and aligned facts are fed back into our QA pipeline by training all models.

Fact alignment: First, we align facts (i. e., relations) from the initial KB with sentences from the text corpus. These sentence-fact tuples can later be used to train the modules during knowledge transfer. For every fact $\langle h, r_j, t\rangle \in \mathcal{E}$, we retrieve the document $d^{(h)} \in \mathcal{D}$ describing h and split it into sentences. Then we filter for those sentences that contain the tail entity t, i. e., contain any of its names $\psi_i^{(t)}$. In cases where there are multiple facts with the same relation type and head entity but different tail entities, we aggregate the facts and keep all sentences that mention at least one tail entity t.[1] Furthermore, we filter for stronger evidence by using the semantic names of the relation. Formally, we first tokenize all names of the relation and the sentences. Second, we remove stop words from them. Third, we calculate the number of overlapping tokens θ_i with each sentence i. We discard all facts that have no sentences with an overlap. For all others, we select the sentence with the highest score θ_i. This approach results in a list of sentence-fact tuples that can be used for fine-tuning by the following knowledge transfer.

Negative training samples: In order to generate negative training samples, we use the following procedure. For every fact $\langle h, r_j, t\rangle \in \mathcal{E}$, we retrieve the document $d^{(h)} \in \mathcal{D}$ describing h and remove all sentences that mention any name $\psi_i^{(t)}$ of the tail entity t. Then we we rank all remaining sentences with our sentence selection module. Finally, we pair the triplet with the highest ranking sentence and label the pair as unanswerable.

Knowledge transfer: Now the above training samples are fed into our QA pipeline. The relation extraction module is fine-tuned by constructing BERT samples analogous to the ones used during prediction. To this end, for every fact-sentence tuple, we randomly select one of the relation names and concatenate it with a question mark, the separation token, and the sentence itself. We then use the first occurrence of a name for any of the tail entities, $\psi_i^{(t)}$, in order to define both the start and end point of the answer. For negative facts, we set the start and end point of the answer to the `"[SEP]"` token.

After having fine-tuned the module (P2) for relation extraction, we can use it in combination with the module for sentence selection (P1) in order to make predictions for known triples in our initial KB. These predictions facilitate the fine-tuning of our re-ranking module. Formally, we proceed as follows: for every fact $\langle h, r_j, t\rangle \in \mathcal{E}$ in our initial KB, we mask the tail entity and issue a query $\langle h, r_j, \_\rangle$ against the existing system. This results in a list of candidate answers for every fact. Since we know the ground-truth tail entity, we can create noisy labels that belong to the answer candidate answers. These labels are noisy because we only compare if the extracted answer matches any name of the tail entities but do not yet link that name back into the KB. Finally, the labeled candidate answers are used to fine-tuning the module (P3) for answer re-ranking.

5.2 Continuous Improvement from User Interactions

Predictions for relationship-types that have not been seen during training (e. g., zero-shot setting) are known to be hard. The inherent benefit of using an interactive approach is that we can generate training data on zero-shot relations on-the-fly during annotation: All facts $\langle h, r_j, t\rangle$ that get approved by users are immediately added to our knowledge base. Simultaneously, we use new facts in order to generate

[1] As an example, let us consider the initial KB containing the triples $\langle$`New York Times`, `award`, `Pulitzer Prize`$\rangle$ and $\langle$`New York Times`, `award`, `George Polk Award`$\rangle$. Then we keep all sentences in $d^{(\texttt{New York Times})}$ that contain any $\psi_i^{(\texttt{Pulitzer Prize})}$ or $\psi_i^{(\texttt{George Polk Award})}$.

positive and negative training samples as described above for fact alignment. Finally, we add the new samples to our training set and trigger model updates (i. e., iterative repetitions of the knowledge transfer step) after a batch of successful annotations.

6 Dataset

Common datasets for KB completion (Mahdisoltani et al., 2014; Toutanova et al., 2015) are designed for link prediction and lack a text corpus from which the task of KB completion could be performed. The WikiReadings dataset (Hewlett et al., 2016) comes close to our needs but 99% of the dataset are covered by only 180 relations. Hence, it largely lacks infrequent long-tail relations that are known to be notoriously hard to predict. Instead, we construct a dataset that is tailored to our setting by following best-practice as described below (and in the supplements).

Our dataset is a combination of (i) a incomplete KB in the form of a subset of Wikidata and (ii) a text corpus, namely the English Wikipedia. Both are detailed below.

KB subsets: Relations in the KB are split into two distinct subsets (for performing different evaluations later): (i) *known* relations represent an initially known and incomplete KB, while (ii) *zero-shot* or cold-start relations are unseen during the initial training. Both subsets are further divided into a training set and a hold-out set for testing.

The different subsets are used at different states of our training and evaluation procedure as follows: The *known training-set* is used during the initial fact alignment and knowledge transfer in order to learn our system. The *known test-set* is used to evaluate the performance gains on known relations when using both fact alignment and knowledge transfer from before. The *zero-shot training-set* is used for simulating user interactions. We remind that, even though this set is called "train", we do not use its ground-truth labels directly but simulate user interactions that approve correct labels and reject incorrect labels. The *zero-shot test-set* is used to evaluate performance on those unseen relations. It allows us to quantify the performance improvements from continuous learning.

Negative examples: We synthetically generate unanswerable examples in order to make the prediction task harder and evaluate the answer triggering module. Therefore, we copy queries in our test-sets and pair them with Wikipedia articles in which all sentences that contain mentions of the correct tail entity have been removed.

7 Computational Experiments

Overview: Our proposed training framework is evaluated as follows: In Sec. 7.1, we first record the performance on the *known* subset of the dataset, i. e., on those relations that were initially present in our KB. In Sec. 7.2, we evaluate the performance on the *zero-shot* subset of our dataset. This represents the initial cold-start performance of our system. Furthermore, in Sec. 7.3, we present our experiment on continuous improvements from over 9000 simulated user interactions. Finally, in Sec. 7.4, we present our results including entity back-linking.

Performance metrics: In the following, our KB completion over free text is evaluated using different performance metrics. We report the hit rate at k, i. e., H@k, when evaluating only on positive examples (e.g., queries that have an answer in the text corpus and hence disabling the answer triggering module for this evaluation). For negative examples (e.g., only queries that are not answerable), we report the negative recall *neg-Rec*, which is calculated from the fraction of negative samples where the framework triggered the fold option. Finally, we report the combined accuracy *comb-Acc* when evaluating on positive and negative samples at the same time. The combined accuracy is 1 for an example iff it was positive and the correct answer is predicted or it was negative and the fold option was triggered, and 0 otherwise.

Baselines: Our proposed KB completion is compared against the following two baselines: (i) BERT-Sentence: This baselines combines the relation extraction module from BERT with our proposed sentence selection. Thereby, it resembles the system from Levy et al. (2017) in the sense that we replaced the machine comprehension model by BERT and the manual question templates replaced by our (noisy) keyword queries. (ii) Naïve QA pipeline: This baseline resembles our pipeline for KB completion including the module for answer re-ranking. However, all modules were simply trained on

the SQuADv2 dataset (Rajpurkar et al., 2018), which includes negative examples. Other baselines lack comparability as they operate on structural information in the KB and not textual content. This is to the best of our knowledge the first verifiable KB completion framework.

7.1 Performance on "Known" Subset

Tab. 1 provides the performance for the "known" relations in our KB by the following breakdown: (i) *all known relations* is listed in the top row; (ii) *frequent* are relations with more than 100 training samples during fact alignment; (iii) *few-shot* have between 2–10 training samples; and (iv) *one-shot* has only a single training sample. Furthermore, we show evaluations for only positive queries, only negative queries and a combined evaluation (see performance metrics).

	Baselines								Our training framework			
	BERT-Sentence				Naïve QA pipeline				After knowledge transfer			
	pos		neg	both	pos		neg	both	pos		neg	both
	H@1	H@5	neg-Rec	comb-Acc	H@1	H@5	neg-Rec	comb-Acc	H@1	H@5	neg-Rec	comb-Acc
All known relations	0.175	0.322	0.225	0.201	0.224	0.343	0.225	0.226	0.602	0.713	0.805	0.602
⊂ frequent relations	0.184	0.337	0.205	0.195	0.232	0.360	0.205	0.219	0.607	0.719	0.885	0.606
⊂ few-shot relations	0.168	0.339	0.505	0.345	0.254	0.346	0.505	0.388	0.337	0.497	0.936	0.567
⊂ one-shot relations	0.065	0.102	0.398	0.245	0.065	0.111	0.398	0.245	0.115	0.154	0.798	0.443

Table 1: Performance (i. e., as measured on the hold-out split) on the "known" subsets of our dataset. The first row reports the performance for all known relations, followed by subsets thereof as described in the text.

Evidently, KB completion with knowledge transfer improves over both baselines substantially. For instance, knowledge transfer increases the hit rate H@1 of the best baseline from 0.224 to 0.602, i. e., a plus of 0.378. As expected, the performance in the case of one-shot relations is lower than for others. For frequent relations, we succeed in answering at least every second relation correctly. The results for all other performance metrics reveal a similar pattern. The performance of our answer-triggering module is similarly high: For four out of five questions that are unanswerable our module triggers the fold option. For the evaluation on both positive and negative part our combined accuracy achieves around 60%.

7.2 Performance on "Zero-Shot" Subset

Tab. 2 states the performance for detecting "zero-shot" relations. That is before any user interactions have occurred and there was no training-data generated from those relations. In detail we show: (i) *all zero-shot relations* as listed in the top row; (ii) *unseen head* is a subset thereof for which the head entity was not included in the "known" subset; (iii) *unseen tail* is the counterpart for tail entities; (iv) *unseen head&tail* is a combination of both; and (v) *infrequent* refers to relations from the long-tail, that is, where there were fewer than five test samples for evaluation. The latter group of infrequent relations is known to challenging to predict (Xiong et al., 2018) and, to the best of our knowledge, our work is the first that attempts zero-shot predictions on such relations.

	Baselines								Our training framework			
	BERT-Sentence				Naïve QA pipeline				After knowledge transfer			
	pos		neg	both	pos		neg	both	pos		neg	both
	H@1	H@5	neg-Rec	comb-Acc	H@1	H@5	neg-Rec	comb-Acc	H@1	H@5	neg-Rec	comb-Acc
All zero-shot relations	0.164	0.333	0.366	0.267	0.227	0.369	0.366	0.298	0.297	0.449	0.766	0.444
⊂ unseen head	0.164	0.331	0.372	0.270	0.227	0.368	0.372	0.301	0.301	0.450	0.768	0.445
⊂ unseen tail	0.175	0.360	0.366	0.296	0.251	0.395	0.366	0.324	0.332	0.482	0.766	0.535
⊂ unseen head&tail	0.176	0.359	0.372	0.300	0.251	0.395	0.372	0.328	0.335	0.483	0.768	0.768
⊂ infrequent	0.0276	0.064	0.449	0.239	0.055	0.083	0.449	0.252	0.147	0.449	0.862	0.436

Table 2: Performance (i. e., as measured on the hold-out split) on the "zero-shot" subsets of our dataset. The first row reports the performance for all known relations, followed by subsets thereof as described in the text.

All baselines are outperformed by KB completion with knowledge transfer. On all zero-shot predic-

tions, the latter achieves of performance where it is correct in approximately one out of three predictions, and a combined accuarcy of around 44%. In comparison, the best baseline is correct only in a mere one out of five predictions. We make further observations. First, the performance is largely on par with subset of unseen head and unseen tail entities. This demonstrates a prime strength of our QA pipeline in comparison to KB completion via link prediction, since the latter would have been impeded by such missing entities. Second, the performance on infrequent relations is fairly low and thus in line with our expectations. This confirms that making such predictions in a zero-shot setting is extremely difficult. Nonetheless, our approach leads to an improvement on infrequent long-tail distributions.

7.3 Continuous Learning from User Interactions

We now run an interactive experiment simulating more than 9k user interactions. We initialize our system as in the experiment above using only the training-set of the *known* part of the KB. Next, we use the training-set of the *zero-shot* subset in order to simulate user interactions. For every relationship-type in that part of the training-set, we make predictions for up to 30 random queries. We then filter those queries where our system activates the fold option and show only those to the user where a user interaction is triggered. To limit user interactions, we show a maximum of 10 queries per relation to simulated human annotators. Since we know the ground-truth answers to queries, we can simulate users who provide positive feedback on correct answers and negative feedback on incorrect answers. After receiving feedback on those queries, we invoke our continuous update mechanism to re-train and improve the framework. After every iteration, we evaluate our framework on the test-split of the zero-shot subset of our dataset. By that, we measure how performance improves over time on initially unseen relationship types.

In Fig. 2 (left), we show the improvement of the Hits@1 ratio when evaluating the positive part of the test set. The Hits@1 ratio of the system improves from 29.7% to 46.2% (a total improvement of 16.5 percentage points). Fig. 2 (center) shows the neg-rec including the answer triggering stage. We see that there is a trad-off in learning to make right predictions and recognizing unanswerable queries. Fig. 2 (right) shows the comb-acc when mixing positive and negative examples for the evaluation and including the answer triggering. Overall we see that the system improves about 14 percent.

We are currently running a live demo of our system, available from `https://wikidatacomplete.org/`.

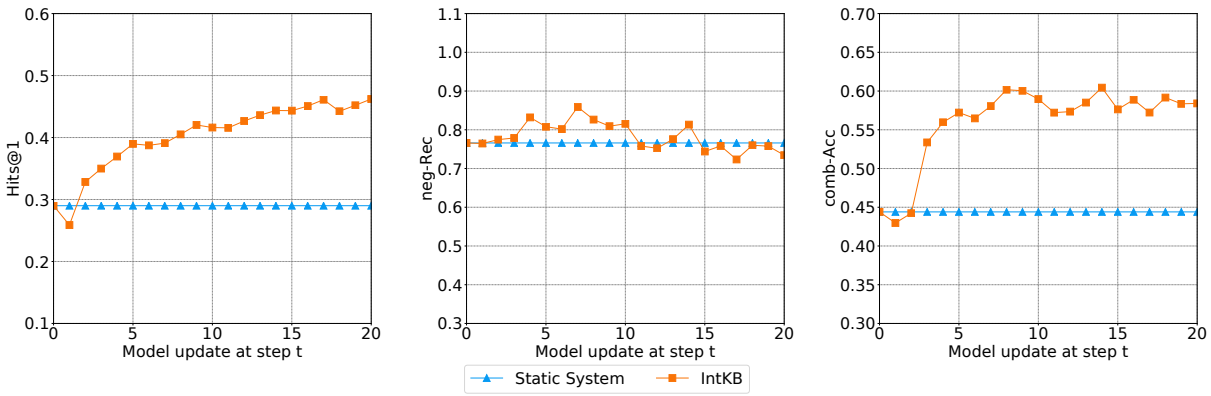

Figure 2: Continuous improvement on initially unseen relations from user interactions.

7.4 Performance on Answer Back-Linking

Tab. 3 states the end-to-end performance including answer back-linking. It thus measures the overall quality of inferring the correct KB entities. In general, we expect that the performance is considerably lower than for text spans. The reason is that, for Wikidata, more than 60% of the text spans are linked to more than one KB entity and, in some cases, match over 100 different KB entities.

For the "known" subset, the correct KB entity is detected with a precision (P@1) of 49.0%. As expected, this value is lower than the H@1 of 0.602 for text spans. The "zero-shot" subset is more challenging but our KB completion still achieves a precision (P@1) of 25.3%. Despite the additional complexity of the task, this value is close to the H@1 of 0.297 for text spans. Finally, the ML-based strategy is largely superior over the other two but can only be used in a supervised setting. The range-based strategy is to strict, often ruling out the correct match.

	Subset: known relations			Subset: zero-shot relations		
Back-linking	P@1	P@2	P@5	P@1	P@2	P@5
Rank-based	0.440	0.500	0.511	**0.253**	**0.273**	**0.276**
Range-based	0.433	0.4692	0.474	0.227	0.238	0.240
ML-based	**0.494**	**0.512**	**0.516**	–	–	–

Table 3: Performance (i. e., as measured on the hold-out test set) of either the "known" or "zero-shot" subset of our dataset across the three different back-linking strategies.

8 Discussion

This paper contributes KB completion using only a single text corpus. For this purpose, we developed a combination of a neural QA pipeline and a novel training framework with fact alignment and contionus updates. Based on it, new facts for incomplete KBs can be inferred effectively.

Our training framework provides a natural approach to align facts with sentences from the corpus and, therefore, makes all prediction verifiable. This is a clear advantage over most of the current KB completion techniques that lack means for new facts to be verified by humans. In practice, it has two main implications: (i) Facts can be shown along textual evidence to KB editors and thereby easily approved or rejected. This thus facilitates a human-in-the-loop framework. (ii) In our work, textual evidence of the proposed facts provides information on the provenance, i. e., the source of facts can be traced. This is important in domains (e. g., medicine), where KBs serve as the basis for high-stakes decisions.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPUs used for this research.

References

Farahnaz Akrami, Mohammed Samiul Saeef, Qingheng Zhang, Wei Hu, and Chengkai Li. 2020. Realistic re-evaluation of knowledge graph completion methods: An experimental study. In *International conference on Management of Data (SIGMOD)*.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web*, pages 722–735, Berlin, Heidelberg. Springer Berlin Heidelberg.

Michael Bendersky, Xuanhui Wang, Donald Metzler, and Marc Najork. 2017. Learning from user interactions in personal search via attribute parameterization. In *ACM International Conference on Web Search and Data Mining (WSDM)*, pages 791–799.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Yang Wang. 2018. Variational knowledge graph reasoning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1823–1832, New Orleans, Louisiana, June. Association for Computational Linguistics.

Mingyang Chen, Wen Zhang, Wei Zhang, Qiang Chen, and Huajun Chen. 2019. Meta relational learning for few-shot link prediction in knowledge graphs. 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)*, pages 4216–4225, Hong Kong, China, November. Association for Computational Linguistics.

Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 365–374. ACM.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *"International Conference on Learning Representations (ICLR)"*.

Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2019. Building dynamic knowledge graphs from text using machine reading comprehension. In *International Conference on Learning Representations ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, oct.

Dennis Diefenbach and Andreas Thalhammer. 2018. Pagerank and generic entity summarization for rdf knowledge bases. In *European Semantic Web Conference*, pages 145–160. Springer.

Dennis Diefenbach, Pedro Henrique Migliatti, Omar Qawasmeh, Vincent Lully, Kamal Singh, and Pierre Maret. 2019. Qanswer: A question answering prototype bridging the gap between a considerable part of the lod cloud and end-users. In *The World Wide Web Conference*, pages 3507–3510. ACM.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM.

Alberto Garcia-Duran and Mathias Niepert. 2017. Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features. *arXiv preprint arXiv:1709.04676*.

Matt Gardner and Tom Mitchell. 2015. Efficient and expressive knowledge base completion using subgraph feature extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1488–1498, Lisbon, Portugal, September. Association for Computational Linguistics.

Matt Gardner, Jonathan Berant, Hannaneh Hajishirzi, Alon Talmor, and Sewon Min. 2019. Question answering is a format; when is it useful? *arXiv preprint arXiv:1909.11291*.

Wenqiang He, Yansong Feng, Lei Zou, and Dongyan Zhao. 2015. Knowledge base completion using matrix factorization. In *Asia-Pacific Web Conference*, pages 256–267. Springer.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. WikiReading: A novel large-scale language understanding task over Wikipedia. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1545, Berlin, Germany, August. Association for Computational Linguistics.

Bernhard Kratzwald and Stefan Feuerriegel. 2019. Learning from on-line user feedback in neural question answering on the web. In *The World Wide Web Conference*, WWW '19, pages 906–916, New York, NY, USA. ACM.

Bernhard Kratzwald, Anna Eigenmann, and Stefan Feuerriegel. 2019. Rankqa: Neural question answering with answer re-ranking. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. *arXiv preprint arXiv:1806.07297*.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529–539. Association for Computational Linguistics.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*.

Farzaneh Mahdisoltani, Joanna Asia Biega, and Fabian M. Suchanek. 2014. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*.

Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8. ACM.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. *arXiv preprint arXiv:1504.06662*.

Altaf Rahman and Vincent Ng. 2011. Coreference resolution with world knowledge. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 814–824. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia, July. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia, June. Association for Computational Linguistics.

Avneesh Saluja, Ian Lane, and Ying Zhang. 2012. Machine translation with binary feedback: a large-margin approach. In *Proceedings of the Tenth Biennial Conference of the Association for Machine Translation in the Americas*.

Josua Stadelmaier and Sebastian Padó. 2019. Modeling paths for explainable knowledge base completion. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 147–157, Florence, Italy, August. Association for Computational Linguistics.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. Yago: A large ontology from wikipedia and wordnet. *Journal of Web Semantics*, 6(3):203–217.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Lisbon, Portugal, September. Association for Computational Linguistics.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Zihao Wang, Kwun Ping Lai, Piji Li, Lidong Bing, and Wai Lam. 2019. Tackling long-tailed relations and uncommon entities in knowledge graph completion. *arXiv preprint arXiv:1909.11359*.

Gerhard Weikum and Martin Theobald. 2010. From information to knowledge: Harvesting entities and relationships from web sources. In *Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '10, pages 65–76, New York, NY, USA. ACM.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*, pages 515–526. ACM.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2659–2665. AAAI Press.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573, Copenhagen, Denmark, September. Association for Computational Linguistics.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-shot relational learning for knowledge graphs. *arXiv preprint arXiv:1808.09040*.

Ziyu Yao, Yiqi Tang, Wen-tau Yih, Huan Sun, and Yu Su. 2020. An imitation game for learning semantic parsers from user interaction. *arXiv preprint arXiv:2005.00689*.

Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 267–272, Lisbon, Portugal, September. Association for Computational Linguistics.

Kaja Zupanc and Jesse Davis. 2018. Estimating rule quality for knowledge base completion with the relationship between coverage assumption. In *Proceedings of the 2018 World Wide Web Conference*, pages 1073–1081. International World Wide Web Conferences Steering Committee.

Appendix

A Details on Dataset Preprocessing

In the following we detail how we select our Dataset.

KB preprocessing: We first sample Wikidata triplets that satisfy all of the following three properties: (i) the head entity corresponds to a Wikipedia article (i. e., this is needed to link KB and text corpus); (ii) the tail entity has a matching Wikidata entity (i. e., it is neither a numerical value nor an external identifier); and (iii) the tail entity is mentioned in the Wikipedia article that belongs to the head entity.

The above set of triplets is further processed: First, we discard all relation types that contain fewer than 10 facts. This is needed to ensure meaningful evaluations. Second, we merge facts that have the same head entity and relation type but different tail entities in order to allow for a correct evaluation. Third, out of the remainder, we sample 300 relations, i. e., 250 serve as known relations and 50 as zero-shot relations. Here sampling considers the relative frequencies of tail entities in the original KB.

Train/test splits: We split facts into a training and test set in a 90:10 ratio, but sample at most 1000 for the training split and 500 for the test split. Relations that contain fewer than 20 facts are split in a 50/50 ratio in order to avoid too few test samples. This is different from WikiReadings, where 99% of the dataset coverage is achieved by 180 properties. Our dataset gives more weight to infrequent long-tail relations. The key statistics of our dataset are summarised in Tab. 4.

Naming: The names $\omega^{(j)}$ for the relations are determined based on the English aliases that are stored in the Wikidata. In detail, we use the five longest aliases present. For the entity names $\psi^{(a)}$, we use all Wikidata aliases stored with the entity.

	Subset: known	Subset: zero-shot
#relations	250	50
#facts (train)	114,797	19,857
#facts (test)	43,272	7,971
#unique head entities	128,863	16,984
#unique tail entities	199,300	19,857

Table 4: Summary statistics of our dataset for Wikidata-based KB completion over free text from Wikipedia.