When a Knowledge Base Is Not Enough: Question Answering over Knowledge Bases with External Text Data

Denis Savenkov
Emory University
dsavenk@emory.edu

Eugene Agichtein
Emory University
eugene@mathcs.emory.edu

ABSTRACT

One of the major challenges for automated question answering over Knowledge Bases (KBQA), is translating a natural language question to the Knowledge Base (KB) entities and predicates. Previous systems have used a limited amount of training data to learn a lexicon that is later used for question answering. This approach does not make use of other potentially relevant text data, outside the KB, which could supplement the available information. We introduce a new system, Text2KB, that enriches question answering over a knowledge base by using external text data. Specifically, we revisit different phases in the KBQA process and demonstrate that text resources improve question interpretation, candidate generation and ranking. Building on a state-of-the-art traditional KBQA system, Text2KB utilizes web search results, community question answering and general text document collection data, to detect question topic entities, map question phrases to KB predicates, and to enrich the features of the candidates derived from the KB. Text2KB significantly improves performance over the baseline KBQA method, as measured on a popular WebQuestions dataset. The results and insights developed in this work can guide future efforts on combining textual and structured KB data for question answering.

1. INTRODUCTION

It has long been recognized that searchers prefer concise and specific answers, rather than lists of document results. In particular, factoid questions have been an active focus of research for decades due to both practical importance and relatively objective evaluation criteria. As an important example, a large proportion of Web search queries are looking for entities or their attributes [19], a setting on which we focus in this work.

Two relatively separate approaches for Question Answering (QA) have emerged: text-centric, or Text-QA and knowledge base-centric, or KBQA. In the more traditional, Text-QA approach, systems use text document collections to retrieve passages relevant to a question and extract candidate answers [14]. Unfortunately, a passage of text provides a limited amount of information about the mentioned entities, which has to be inferred from the context. The KBQA approach, which evolved from the database community, relies on large scale knowledge bases, such as DBpedia [1], Freebase [9], WikiData [24] and others, which store a vast amount of general knowledge about different kinds of entities. This information, encoded as `[subject, predicate, object]` RDF triples, can be effectively queried using structured query languages, such as SPARQL.

Both approaches eventually deal with natural language questions, in which information needs are expressed by the users. While question understanding is difficult in itself, this setting is particularly challenging for KBQA systems, as it requires a translation of a text question into a structured query language, which is complicated because of the complexity of a KB schema, and many differences between natural language and knowledge representations. For example, Figure 1 shows a SPARQL query that retrieves the answer to a relatively simple question *"who was the president of the Dominican Republic in 2010?"* from Freebase.

KBQA systems must address three challenges, namely question entity identification (to anchor the query process); candidate answer generation; and candidate ranking. We will show that these challenges can be alleviated by the appropriate use of external textual data. Entity identification seeds the answer search process, and therefore the performance of the whole system greatly depends on this stage [28]. Question text is often quite short, may contain typos and other problems, that complicate entity linking. Existing approaches are usually based on dictionaries that contain entity names, aliases and some other phrases, used to refer to the entities [21]. These dictionaries are noisy and incomplete, *e.g.,* to answer the question *"what year did tut became king?"* a system needs to detect a mention *"tut"*, which refers to the entity `Tutankhamun`. If a dictionary doesn't contain a mapping *"tut"* → `Tutankhamun`, as happens for one of the state of the art systems, it will not be able to answer the question correctly. Such less popular name variations are often used along with full names inside text documents, for example, to avoid repetitions. Therefore, we propose to look into web search results to find variations of question entity names, which can be easier to link to a KB (Figure 2). This idea has been shown effective in entity linking for web search queries[1] [12].

After question entities have been identified, answer candidates need to be generated and ranked to select the best answer. A candidate query includes one or multiple triple pat-

SIGIR '16, July 17-21, 2016, Pisa, Italy

DOI: http://dx.doi.org/10.1145/2911451.2911536

[1] http://web-ngram.research.microsoft.com/ERD2014/

```
SELECT DISTINCT ?name {
    :m.027rn :government.governmental_jurisdiction.governing_officials ?gov_position .
    ?gov_position :government.government_position_held.basic_title :m.060c4 .
    ?gov_position :government.government_position_held.office_holder ?president .
    ?gov_position :government.government_position_held.from ?from_date .
    ?gov_position :government.government_position_held.to ?to_date .
    FILTER ( xsd:date(?from_date) <= "2010"^^xsd:date AND xsd:date(?to_date) >= "2010"^^xsd:date)
    ?president :type.object.name ?name
}
```

Figure 1: SPARQL query to retrieve the answer to the question *"who was the president of the dominican republic in 2010?"* from Freebase

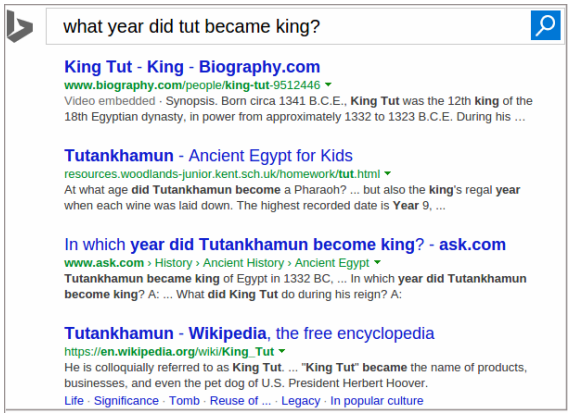

Figure 2: Search results for the question *"what year did tut became king?"*, which mention both the full name of the king and the correct answer to the question

terns with predicates, corresponding to words and phrases in the question. Existing knowledge base question answering approaches [3, 6, 7, 8, 10, 29] rely on a lexicon, learned from manually labeled training data, and supported by additional resources, such as question paraphrases [7] and weakly labeled sentences from a large text collection [30]. Such training data tends to be small compared to the number of different predicates in a KB, and therefore the coverage of these lexicons is limited. By our estimate, in a popular WebQuestions KBQA dataset, the answers to ∼5.5% of test questions (112 out of 2032) involve a predicate that does not appear as a ground truth in the training set. For example, an RDF triple `[Bigos, food.dish.type_of_dish1, Stew]` answers the question *"what are bigos?"*, but no other examples in the training set involve this predicate. In addition, a lexicon needs to cover all different ways a predicate can be asked about. For example, questions *"who did jon gosselin cheat with?"* and *"who is the woman that john edwards had an affair with?"* are answered by the same KB predicate, but use different language. Therefore, presence of the first question in a training set may not help to answer the second question. On the other hand, traditional Text-QA systems benefit from the redundancy of the information on the Web, where the same facts are stated multiple times in many different ways [17]. This increases the chances of a good lexical match between a question and answer statements, which makes even some relatively simple counting-based techniques quite effective [11]. We propose to adapt these ideas from text-based question answering for KBQA. The right part of the Figure 3 shows web search results, a community question answering page, and text fragments mentioning pairs of entities, that can be useful to answer the question about John Edwards' affair.

To summarize, our contributions are three-fold:

- A novel "hybrid" knowledge base question answering system, which uses both structured data from a knowledge base and unstructured text resources. Section 3 describes the architecture of our system, and Section 4 shows that this fusion improves the performance of a state of the art KBQA system.
- Novel data sources and techniques for KBQA: enhancing question entity identification using web search results (Section 3.1); improving predicate matching by mining CQA data (Section 3.2); and improving candidate ranking by incorporating text corpus statistics (Section 3.3).
- Comprehensive empirical analysis of our system on a popular WebQuestions benchmark, demonstrating that additional text resources can improve the performance of a state-of-the-art KBQA system (Section 4). In addition, we conduct an extensive analysis of the system to identify promising directions for future improvements (Section 5).

Taken together, this work introduces novel techniques for using external text to significantly improve the performance of the KBQA approach. More broadly, our work bridges the gap between Text-QA and KBQA worlds, demonstrating an important step forward towards combining unstructured and structured data for question answering.

2. OVERVIEW OF KNOWLEDGE BASE QUESTION ANSWERING

In this section, we overview the existing approaches to Knowledge Base Question Answering (KBQA), as our approach, described in the next section, builds upon and extends some of these efforts.

Over time, KBQA systems have converged to two major approaches: *semantic parsing*, and *information extraction* (IE) [29]. The former focuses on question understanding, and attempts to parse sentences into their semantic representations, *e.g.,* logical forms [6, 7, 8]. IE approaches [3, 31, 30] are based on identifying topic entities in the question, and then, using pre-defined templates for mapping the question to predicates, explore the neighborhood of these entities in a KB. Theoretically, semantic parsing-based systems would be capable of generating any required queries, and would apply to any question, seen or unseen in training, whereas the template-based approach is less likely to generalize. In practice, however, answers to most of the questions lie within two edge traversals in a KB, making the template-based approaches quite effective.

Recent resurgence of interest in KBQA coincides with the availability of large scale knowledge bases such as Freebase and DBPedia, as well as commercial efforts from Google, Microsoft, Facebook and Yahoo; which make it possible to answer many real questions. Additionally, the creation of the WebQuestions dataset [6], provided a common benchmark which is large enough to allow both comprehensive evaluation, and training machine learning methods. In this work, we chose to extend an existing information extraction KBQA system – Aqqu [3] – which achieves one of the highest scores among publicly available systems. However, as we will show, our approach is general and can be incorporated into other IE-based systems as well.

We will first describe an information extraction approach to KBQA in more detail using Aqqu as an example. In Section 3 we present our system Text2KB, which extends this approach by incorporating external text-based data at various stages of the question answering process.

2.1 The Aqqu KBQA system

First, the system identifies question entities, which are used as sources for the answer search process. For concreteness, consider a question from the WebQuestions dataset *"who is the woman that john edwards had an affair with?"*. Here, the entity `John Edwards` with Freebase id `/m/01651q` is the main question entity. However, Freebase contains millions of entities and it can be difficult to identify the topical ones (*e.g.,* entities `Woman` and `Affair` are also present in Freebase), or to disambiguate and choose between `John Edwards` a politician (`/m/01641q`), an American racing driver (`/m/06zs089`) and other people with the same name. Aqqu considers all spans of question words under certain conditions on part of speech tags and uses an entity names lexicon [21] to map phrases to potential entities. Most reported systems, including Aqqu, do not disambiguate entities at this stage, but rather keep a set of candidates along with some information about their popularities (*e.g.,* number of mentions in the collection), and mention scores $p(entity|mention\ text)$.

At the next stage, SPARQL query candidates are generated by exploring the neighborhood of the question topic entities using a predefined set of query templates. Each query template has question entities, predicates and answer placeholders. The majority of the answers in the WebQuestions dataset can be covered by just 3 templates (q_entity - question entity, a_entity - answer entity, cvt_node - Freebase mediator node, which represent tuples with more than 2 arguments):

```
SELECT DISTINCT ?a_entity {
   <q_entity> <predicate> ?a_entity .
}
```

```
SELECT DISTINCT ?a_entity {
   <q_entity> <predicate_1> ?cvt_node .
   ?cvt_node <predicate_2> ?a_entity .
}
```

```
SELECT DISTINCT ?a_entity {
   <q_entity_1> <predicate_1> ?cvt_node .
   ?cvt_node <predicate_2> <q_entity_2> .
   ?cvt_node <predicate_3> ?a_entity .
}
```

The first template retrieves a set of entities that are directly connected to the given question entity via a certain predicate. The second template accounts for the presence of a mediator node, that groups together arguments of a multi-argument relation. And the last template looks for cases, when a question also mentions another argument of a multi-argument relation, *e.g.,* `Captain Kirk` and `Star Trek` for the question *"who played captain kirk in star trek movie?"*.

Each query candidate is represented with a set of features, that includes the scores for linked question entities, various scores for matching between question term n-grams and query predicates, the size of the results list, *etc.* The final stage of the question answering process is filtering and ranking. The Aqqu system employs a pairwise learning-to-rank model, trained on part of the dataset. For each pair of candidate answers Aqqu creates an instance, which contains 3 groups of features: features of the first, the second candidate in the pair and the differences between the corresponding features of the candidates. Specifically, a Random Forest model is used in the provided Aqqu implementation. A pair where the first candidate is better than the second belongs to class +1, and -1 otherwise. To reduce the number of pairs for the final ranking, Aqqu includes a simplified linear filtering model, which is trained to detect incorrect answers with high precision.

2.2 Basic system extensions

Before introducing our text-based improvements, we describe some basic extensions to the original Aqqu system. First, we noticed that since Aqqu does not use information about the answer entity Freebase types, in many cases it returns an answer that is incompatible with the question: *e.g.,* state instead of county *etc.* Therefore, we a trained a model to return a score which measures compatibility between the question and answer entities, based on the entity notable types and question uni- and bi-grams as features, similar to Aqqu's relations score model. A second extension introduced a new date range query template, which helps solve cases like *"what team did david beckham play for in 2011?"*, where we need to look at the ranges of dates to determine whether an answer candidate satisfies the question.

```
SELECT DISTINCT ?a_entity {
   <q_entity_1> <predicate_1> ?cvt_node .
   ?cvt_node <from_predicate> ?date_from .
   ?cvt_node <to_predicate> ?date_to .
   ?cvt_node <predicate_2> ?a_entity .
   FILTER ( <question_date> >= ?date_from AND
            <question_date> <= ?date_to )
}
```

3. TEXT2KB: INCORPORATING TEXT DATA INTO KBQA

We now introduce our system, called Text2KB[2], that expands upon the basic KBQA model by incorporating external textual sources throughout the QA process. The general architecture and an example use case of Text2KB is presented on Figure 3. The left part of the figure roughly corresponds to the architecture of existing information extraction approaches to KBQA. The right part introduces additional external text data sources, namely Web search results, community question answering (CQA) data, and a collection of documents with detected KB entity mentions. We demonstrate how these data sources can help with the main challenges in KBQA, *i.e.,* question topical entity identification, predicate scoring and answer candidates ranking.

[2]http://ir.mathcs.emory.edu/projects/text2kb/

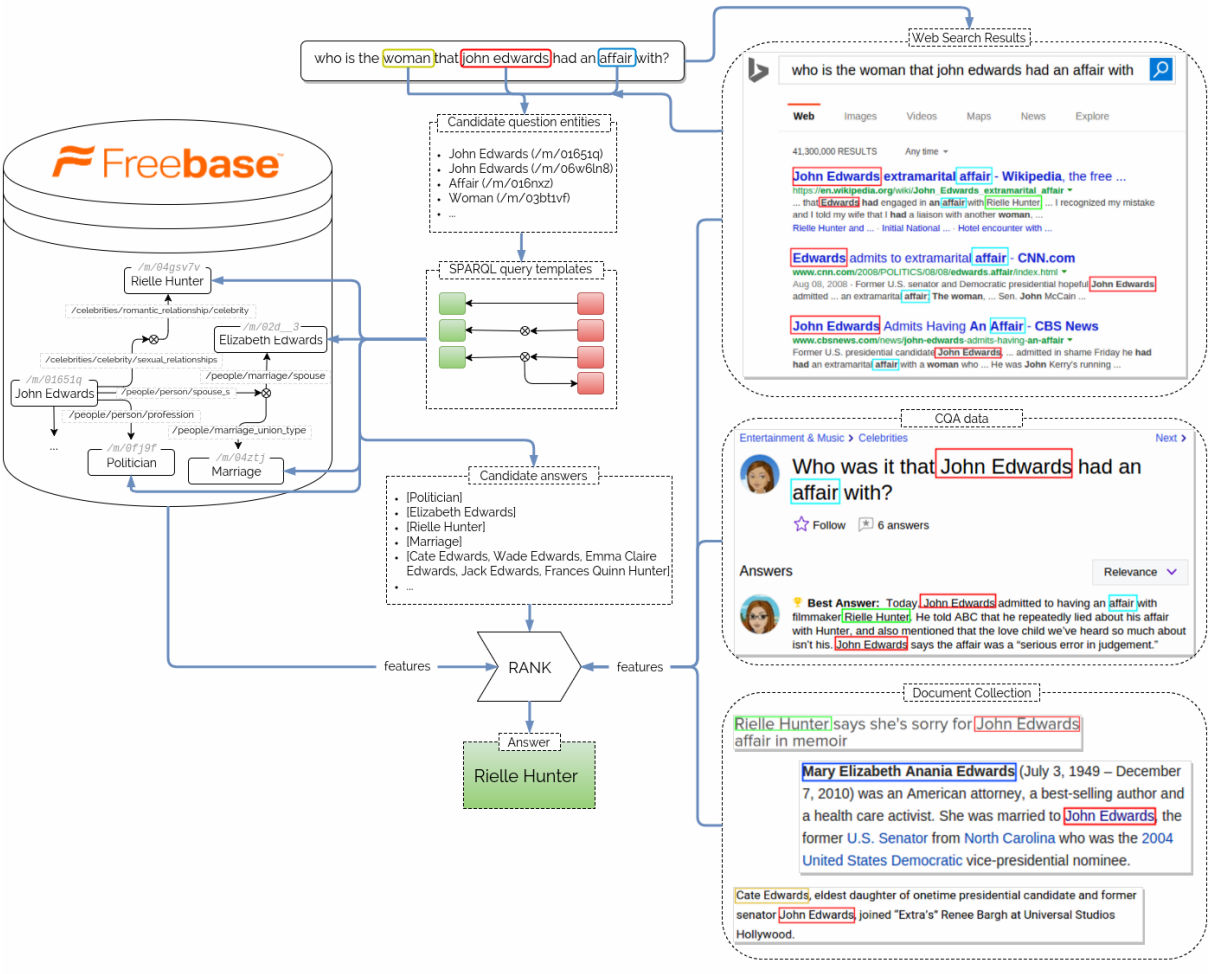

Figure 3: The architecture of our Text2KB Question Answering system

As described in detail next, this information is used to develop novel features for detecting entities and ranking candidate answers. The final ranking is performed using the same learning-to-rank method as the baseline Aqqu system [3], which uses the Random Forest model.

3.1 Web search results for KBQA

Traditional Text-QA systems rely on search results to retrieve relevant documents, which are then used to extract answers to users' questions. Relevant search results mention question entities multiple times and in various forms, which can be helpful for entity linking [12]. Furthermore, retrieved document set often contains multiple statements of the answer, which can be a strong signal for candidate ranking [17].

To obtain related web search results, Text2KB issues the question as a query to a search engine[3], extracts top 10 result snippets and the corresponding documents. Next, Text2KB uses Aqqu entity linking module to detect KB entity mentions in both snippets and documents.

Question entity identification. Question text provides only a limited context for entity disambiguation and linking; additionally, the entity name can be misspelled or an uncommon variation used. This complicates the task of entity identification, which is the foundation of KB question answering process. Fortunately, web search results help with these problems, as they usually contain multiple mentions of the same entities and provide more context for disambiguation. Text2KB uses the search result snippets to *expand* the set of detected question entities. More specifically, we count the frequencies of each entity mentioned in search snippets, and most popular ones with names similar to some of the question terms are added to the list of topical entities. The goal of this similarity condition is to keep only entities that are likely mentioned in the question text, and filter out related, but different entities. To estimate the similarity between a name and question tokens, we use Jaro-Winkler string distance. An entity is added to the list of question entities if at least one of its tokens e_t has high similarity with one of the question tokens q_t excluding stopwords ($Stop$):

$$\max_{e_t \in M \setminus Stop, q_t \in Q \setminus Stop} 1 - dist(e_t, q_t) \geq 0.8$$

Answer candidate features. The information stored in KBs can also be present in other formats, *e.g.,* text statements. For example, on Figure 2 multiple search snippets mention the date when Tutankhamun became a king. Text-QA systems use such passages to extract answer to users' questions. However, text may not provide sufficient context information about the mentioned entities, and systems have to infer the useful details, *e.g.,* entity types, which can be problematic [31]. On the other hand, KBQA systems can

[3]In our experiments we use the Bing Web Search API https://datamarket.azure.com/dataset/bing/search and local Wikipedia search using Lucene

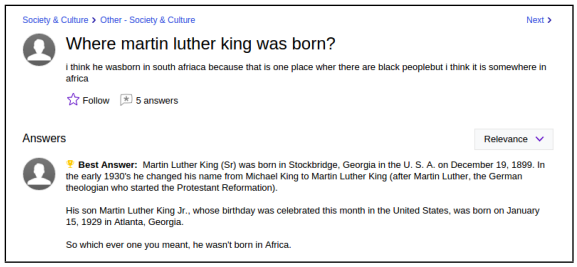

Figure 4: Example of a question and answer pair from Yahoo! Answers CQA website

utilize all the available KB knowledge about the entities in a candidate answer, and would benefit from additional text-based information to improve ranking. More specifically, Text2KB proceeds as follows:

1. Precompute term and entity IDFs. We used Google n-grams corpus to approximate terms IDF by collection frequencies and available ClueWeb Freebase entity annotations[4] to compute entity IDF scores.
2. Each snippet s_i and document d_i are represented by two TF-IDF vectors of lowercased tokens (t_{s_i} and t_{d_i}) and mentioned entities (e_{s_i} and e_{d_i}).
3. In addition, vectors of all snippets and all documents are merged together to form combined token and entity vectors: $t_{\cup s_i}$, $t_{\cup d_i}$, $e_{\cup s_i}$ and $e_{\cup d_i}$.
4. Each answer candidate a_j is also represented as TF-IDF vector of terms (from entity names), and entities: t_{a_j} and ea_j
5. Cosine similarities between answer and each of 10 snippet and document vectors are computed: $\cos(t_{s_i}, t_{a_j})$, $\cos(t_{d_i}, t_{a_j})$ and $\cos(e_{s_i}, e_{a_j})$, $\cos(e_{d_i}, e_{a_j})$. We use the average score and the maximum score as features.
6. We also compute answer similarities with the combined snippet and document vectors: $\cos(t_{\cup s_i}, t_{a_j})$, $\cos(e_{\cup s_i}, e_{a_j})$, $\cos(t_{\cup d_i}, t_{a_j})$, $\cos(e_{\cup d_i}, e_{a_j})$.

3.2 CQA data for Matching Questions to Predicates

Recall that a major challenge in KBQA is that natural language questions do not easily map to entities and predicates in a KB. An established approach for this task is supervised machine learning, which requires labeled examples of questions and the corresponding answers to learn this mapping, which can be expensive to construct. Researchers have proposed to use weakly supervised methods to extend a lexicon with mappings learned from *single sentence statements* mentioning entity pairs in a large corpus [30]. However, the language used in questions to query about a certain predicate may differ from the language used in statements. A recent work [20] demonstrated how distant supervision can be applied to question-answer pairs from CQA archives for a related task of information extraction for knowledge base completion. In a similar way, we use weakly labeled collection of question-answer pairs to compute *associations* between question terms and predicates to *extend* system's lexicon (Figure 4). We emphasize that this data does not replace the mappings learned from single sentence statements, which are already used by our baseline system, but rather introduces the new ones learned from the CQA data.

For our experiments we use 4.4M questions from Yahoo!

[4]http://lemurproject.org/clueweb09/FACC1/

Term	Predicate	PMI score
born	people.person.date_of_birth	3.67
	people.person.date_of_death	2.73
	location.location.people_born_here	1.60
kill	people.deceased_person.cause_of_death	1.70
	book.book.characters	1.55
currency	location.country.currency_formerly_used	5.55
	location.country.currency_used	3.54
school	education.school.school_district	4.14
	people.education.institution	1.70
	sports.school_sports_team.school	1.69
win	sports.sports_team.championships	4.11
	sports.sports_league.championship	3.79

Table 1: Examples of term-predicate pairs with high PMI scores, computed using distant supervision from a CQA collection

WebScope L6 dataset[5]. Question and answer texts were run through an entity linker, that detected mentions of Freebase entities. Next, we use distant supervision assumption to label each question-answer pair with predicates between entities mentioned in the question and in the answer. This labels are used to learn associations between question terms and predicates by computing pointwise mutual information scores (PMI) for each term-predicate pair. Examples of scores for some terms are given in Table 1.

In Text2KB we evaluate candidate answer predicates by using the association (e.g., PMI) scores between predicates and the question terms (missing pairs are given a score of 0). The minimum, average and maximum of these values are used as features to represent a candidate answer. Such associations data can be sparse, we also use pretrained word2vec word embeddings[6]. We compute predicate embeddings by taking a weighted average of term vectors from predicate's PMI table. Each term vector is weighted by its PMI value (terms with negative score are skipped). Then, we compute cosine similarities between predicate vector and each of the question term vectors and take their minimum, average, maximum as features. Finally, we average embeddings of question terms and compute its cosine similarity with the predicate vector.

3.3 Estimating Entity Associations

A key step for ranking candidate answers is to estimate whether the question and answer entities are related in a way asked in the question. Existing KBQA approaches usually focus on scoring the mappings between question phrases and KB concepts from a candidate SPARQL query. However, textual data can provide another angle on the problem, as question and answer entities are likely to be mentioned together somewhere in text passages. For example, in the bottom right corner of Figure 3 we can see some passages that mention a pair of people, and the context of these mentions explains the nature of the relationships. This data can be viewed as additional edges in a KB, which connect pairs of entities, and have associated language models, estimated from text phrases, that mention these entities. Such edges do not have to coincide with the existing KB edges, and can connect arbitrary pairs of entities, that are mentioned together in text, therefore extending the KB.

[5]https://webscope.sandbox.yahoo.com/catalog.php?datatype=l
[6]https://code.google.com/p/word2vec/

Entity 1	Entity 2	Term counts
John Edwards	Rielle Hunter	campaign, affair, mistress, child, former ...
John Edwards	Cate Edwards	daughter, former, senator, courthouse, greensboro, eldest ...
John Edwards	Elizabeth Edwards	wife, hunter, campaign, affair, cancer, rielle, husband ...
John Edwards	Frances Quinn	daughter, john, rielle, father, child, former, paternity...

Table 2: Example of entity pairs along with the most popular terms mentioned around the entities

We use the ClueWeb12 corpus with existing Freebase entity annotations and count different terms that occur in the context of a mention of a pair of different entities (we only consider mentions within 200 characters of each other). To compute this unigram language model we use the terms separating the entities, as well as the terms within a small window (e.g., 100 characters) before and after the entity mentions. A small sample of this data is presented in Table 2.

We use this data to compute candidate ranking features as follows. Consider question words Q and an answer candidate, which contains a question entity e_1 and one or more answer entities e_2. For each answer candidate, we compute a language model score:

$$p(Q|e_1, e_2) = \prod_{t \in Q} p(t|e_1, e_2)$$

and use the minimum, average and maximum over all answer entities as features. To address the sparsity problem, we again use embeddings, *i.e.,* for each entity pair a weighted (by counts) average embedding vector of terms is computed and minimum, average and maximum cosine similarities between these vectors and question token embeddings are used as features.

3.4 Internal text data to enrich entity representation

In addition to external text data, many knowledge bases, including Freebase, contain text data as well, *e.g.,* Freebase includes a description paragraph from Wikipedia for many of its entities. These text fragments provide a general description of entities, which may include information relevant to the question [22]. For completeness, we include them in our system as well. Each entity description is represented by a vector of tokens, and a vector of mentioned entities. We compute cosine similarities between token and entity vectors of the question and description of each of the answers, and use the minimum, average and maximum of the scores as features.

4. EXPERIMENTAL RESULTS

This section reports the experimental setup, including the dataset and metrics, as well as the main methods compared for evaluating the performance of our Text2KB system. Additionally, we describe a series of ablation studies to analyze contribution of different system components.

4.1 Methods Compared

We compare our system, Text2KB, to state-of-the-art approaches, notably:

- **Aqqu**: the state-of-the-art baseline KBQA system [3], described in Section 2.
- **Text2KB(Web search)**: Our Text2KB system, using the Bing search engine API over the Web.
- **Text2KB(Wikipedia search)**: Our Text2KB system, using the standard Lucene search engine over the February 2016 snapshot of the English Wikipedia, in order to validate our system without the potential "black-box" effects of relying on a commercial Web search engine (Bing) and changing corpus (Web).
- **STAGG**: The current highest performing KBQA system [31] as measured on the WebQuestions dataset.

Additionally, results from other previously published results on WebQuestions are included to provide context for the improvements introduced by our Text2KB system.

4.2 Datasets

We followed the standard evaluation procedure for the WebQuestions dataset, and used the original 70-30% train-test split (3,778 training and 2,032 test instances). Within the training split, 10% was set aside for validation to tune the model parameters, described below and only the best - performing set of parameters selected on the validation data was used to report the results on the official test split.

4.3 Evaluation Metrics

Recent papers using the WebQuestions dataset have primarily used the F1-score as the main evaluation metric, defined as:

$$f1(a^*, a) = 2\frac{precision(a^*, a) recall(a^*, a)}{precision(a^*, a) + recall(a^*, a)}$$

where $precision(a^*, a) = \frac{|a^* \cap a|}{|a|}$ and $recall(a^*, a) = \frac{|a^* \cap a|}{|a^*|}$, a^* and a are correct and given answers, which can be lists of entities. Additionally, we report average precision and recall, to gain better understanding of the tradeoffs achieved by different methods.

4.4 Main Results

The results of existing approaches and our Text2KB system are presented in Table 4. We should note, that text-based QA systems typically return a ranked list of answers, whereas many answers on WebQuestions dataset are lists, which complicates the comparison between KBQA and text-based systems. The result reported for YodaQA system is F1 score at position 1.

As we can see, Text2KB significantly improves over the baseline system and reaches the current best published result - STAGG [31]. We believe that this system will also benefit from the ideas of our work (Section 5).

4.5 Datasource and Features Contribution

To analyze the contribution of the features and datasources we introduced, we report results from a series of ablation studies. For convenience, we introduce the following shorthand notations for different components of our system:

- `T` - notable type score model as a ranking feature
- `DF` - date range filter-based query template
- `WebEnt` - using web search result snippets for question entity identification
- `WikiEnt` - using wikipedia search result snippets for question entity identification
- `Web` - using web search results for feature generation

System	avg Recall	avg Precision	F1 of avg P and R	avg F1
OpenQA [16]	-	-	-	0.35
YodaQA [4]	-	-	-	0.343
Jacana [30]	0.458	0.517	0.486	0.330
SemPre [6]	0.413	0.480	0.444	0.357
Subgraph Embeddings [10]	-	-	0.432	0.392
ParaSemPre [7]	0.466	0.405	0.433	0.399
Kitt AI [28]	0.545	0.526	0.535	0.443
AgendaIL [8]	0.557	0.505	0.530	0.497
STAGG [31]	0.607	**0.528**	**0.565**	**0.525**
Aqqu (baseline) [3]	0.604	0.498	0.546	0.494
Text2KB (Wikipedia search)	**0.632*** (+4.6%)	0.498	0.557* (+2.0%)	0.514* (+4.0%)
Text2KB (Web search)	**0.635*** (+5.1%)	0.506* (+1.6%)	**0.563*** (+3.1%)	**0.522*** (+5.7%)

Table 3: Performance of the Text2KB system on WebQuestions dataset compared to the existing approaches. The differences of scores marked * from the baseline Aqqu system are significant with p-value < 0.01

- `Wiki` - using wikipedia search results for feature generation
- `CQA` - using CQA-based `[question term, KB predicate]` PMI scores for feature generation
- `CW` - features, computed from entity pairs language model, estimated on ClueWeb

In our results table we will use the notation `+<comp>` for a system with a certain component added, and `-<comp>` when it is removed. For example, the baseline system will be denoted as "`Aqqu`". The same system with additional date range filter query templates and notable types score model is denoted as "`Aqqu +DF+T`", which represents the same system as "`Text2KB -WebEnt-Web-CQA-CL`" (we will call it Text2KB (base)). Our full system "`Text2KB`" can be also denoted as "`Aqqu +DF+T+WebEnt+Web+CQA+CL`".

Components: First, we analyze the improvements introduced by different components of our system (Table 4). As we can see, additional date range filters and notable types model (`Aqqu+DF+T`) are responsible for an increased recall and a drop in precision compared to the baseline model. Features generated from Wikipedia search results, CQA data and ClueWeb entity pair language models (`+Wiki+CQA+CL`) improve average F1 by 0.007 (+1.4%) compared to the base model, adding entity linking using Wikipedia search results improves results even more (+3%).

Web search results (`+Web+CQA+CL`) turned out to be more helpful than Wikipedia results (`+Wiki+CQA+CL`), which is natural since Wikipedia is a subset of the web. This was one of the reasons we didn't combine Wikipedia and Web search together. Finally, entity linking and all text-based features combined achieves an even higher score, proving that their contributions are independent.

Data Sources: We now anylize the contribution of the different data sources. We will remove a group of web search, CQA or Clueweb-based features and see how the performance of the whole system changes (Table 5). As we can see, all data sources have an impact on the system performance, and web search results based features provide the most useful signal for answer ranking.

Feature Importance for Ranking: Figure 5 plots a subset of features ranked by their Gini index-based importance scores. The figure supports the observation that web search results features are the most useful, however, other text data sources also contribute to the improvement.

In summary, Text2KB significantly outperforms the baseline system, and each of the introduced components contributes to this improvement. Web search results data turned out to be the most useful resource, and it significantly improves the quality by helping with question entity identification and candidate ranking. Next, we analyze the system performance in more detail, and investigate factors for future extension.

System	R	P	F1
`Aqqu`	0.604	0.498	0.494
`Text2KB (base) = Aqqu+DF+T`	0.617	0.481	0.499
`+Wiki+CQA+CL`	0.623	0.487	0.506
`+WikiEnt +Wiki+CQA+CL`	0.632	0.498	0.514
`+WebEnt`	0.627	0.492	0.508
`+Web+CQA+CL`	0.634	0.497	0.514
`+WebEnt +Web+CQA+CL`	0.635	0.506	0.522

Table 4: Average Recall (R), Precision (P), and F1 of Aqqu and Text2KB system with and without different components. +A means that a component A is added to the Text2KB (base) system. The list of components is given in Section 4.5.

System	R	P	F1
Text2KB (Web search)	0.635	0.506	0.522
`Text2KB -Web`	0.633	0.496	0.513
`Text2KB -CQA`	0.642	0.499	0.519
`Text2KB -CL`	0.644	0.505	0.523
`Text2KB -CQA-CL`	0.642	0.503	0.522
`Text2KB -Web-CQA`	0.631	0.498	0.514
`Text2KB -Web-CL`	0.622	0.493	0.508

Table 5: Average Recall (R), Precision (P), and F1 of Text2KB with and without features based on web search results, CQA data and ClueWeb collection.

5. ANALYSIS AND DISCUSSION

We now investigate how our system would compare to other systems on the same benchmark; then, we investigate in depth the different error modes (Section 5.1), which helps identify the areas of most substantial future improvements.

We took an existing KBQA systems and demonstrated that by combining evidence from knowledge base and external text resources we can boost the performance. A reasonable question is whether the same approach will be helpful to other systems, *e.g.,* the currently best system – STAGG [31]. STAGG differs from our baseline system Aqqu in the components: entity linking algorithm, a set of query templates and

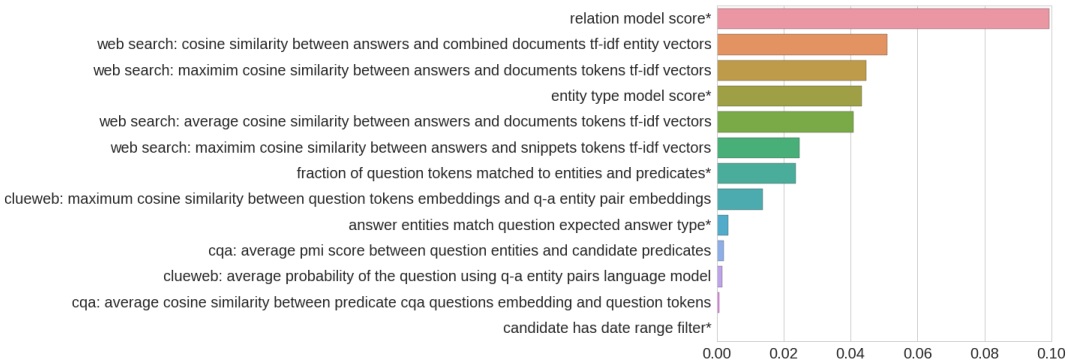

Figure 5: A plot of Gini importances of different features of our answer ranking random forest model (features marked * are not text-based and are provided for comparison)

System	F1
STAGG [31]	0.525
Text2KB + STAGG	0.532 (+1.3 %)
Text2KB + STAGG (Oracle)	0.606 (+15.4 %)

Table 6: Average F1 for combinations of Text2KB and STAGG using a simple heuristic based on the length of the answer list and Oracle upper bound

ranking methods. Therefore, our approach is complementary and should be helpful for STAGG as well. To support this claim, we made an experiment to combine answers of STAGG and Text2KB. One of the advantages of the former is its set of filters, that restricts list results to entities of certain type, gender, *etc.* Therefore, we combined answers of STAGG and Text2KB using a simple heuristic: we chose to use the answer returned by STAGG if the number of answer entities is less than in the Text2KB answer, otherwise we use the answer of our approach. Table 6 gives the results of the experiment, and as we can see the combination achieves a slightly better average F1 score. Alternatively, we can look at the Oracle combination of the systems, which always selects the answer with the higher F1. As we can see such a combination results in a performance of 0.6056, which is much higher than either of the systems.

As we mentioned earlier, answers to 112 of the test questions in the WebQuestions dataset involve predicates that weren't observed in the training set, which may be a problem for approaches that rely on a trained lexicon. We evaluated both systems on these questions, and indeed the performance is very low, *i.e.,* the average F1 score of Text2KB is 0.1640 compared to 0.1199 for STAGG[7].

5.1 Error analysis

To get a better insights into the problems that remain, we collected 1219 questions for which Text2KB didn't return completely correct answer, *i.e.,* F1 score < 1. We manually looked through a couple of hundreds of these examples and grouped the problems into several clusters (Figure 6).

As we can see candidate ranking is still the major problem, and it accounts for $\sim$ 31% of the cases. The second problem is incorrect ground truth labels (almost 25% of reported errors). Another set of questions has incomplete or overcomplete ground truth answer list. Typical examples are questions asking for a list of movies, books, landmarks,

[7]Unfortunately, the number of questions is too low to show statistical significance (p-value=0.16) of the difference

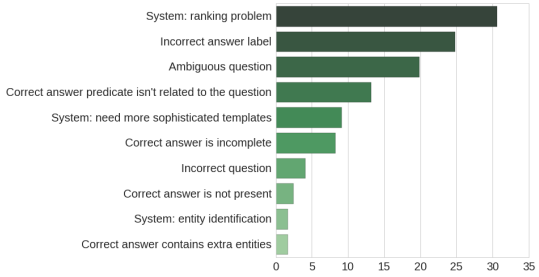

Figure 6: Distribution of problems with questions, where Text2KB returns an answer with F1<1

etc. The ground truth answer usually contains $\sim$ 10 entities, whereas the full list is often much larger. This seems to be an artifact of the labeling process, where the answer was selected from the Freebase entity profile page, which shows only a sample of 10 entities, while the rest are hidden behind the "N values total" link. About 20% of the questions are ambiguous, *i.e.,* questions have no strict 1-1 correspondence with any of the predicates and can be answered by multiple ones without any obvious preferences. For example, the question *"what did hayes do?"* can be answered by profession, occupied position or some other achievements. Another problem is when there is no predicate that answers the question. For example, the question *"what do people in france like to do for fun?"* doesn't have a good match among the facts stored in Freebase. The ground truth entity `Cycling` comes from the list Olympic sport competitions country participated[8].

Text2KB components were quite effective in resolving some of the problems. Web search results helped identify the right question topical entity in a number of cases, *e.g., "what did romo do?"* mentions only the last name of the Dallas Cowboys quarterback and the baseline system were unable to map it to the right entity. Web search results provides more than enough evidence that "*romo*" refers to `Tony Romo`. However, there are a number of loses, introduced by added unrelated entities. For example, the entity `I Love Lucy` was added for the question *"what was lucille ball?"*, because the term *lucy* had high similarity with *lucille*. A portion of these problems can be fixed by a better entity linking strategy, *e.g.,* [12]. An interesting example, when external text resources improved the performance is the question *"what ship did darwin sail around the world?"*. This is actually a hard question, because the ship entity is connected to

[8]`olympics.olympic_participating_country.athletes`

the `Charles Darwin` entity through the "knownFor" predicate along with some other entities like `Natural selection`. Thus, the predicate itself isn't related to the question, but nevertheless, the name of the ship `HMS Beagle` is mentioned multiple times in the web search results, and entity pair model computed from ClueWeb also has high scores for the terms "ship" and "world".

There are several major reasons for the loses, introduced by features based on external text resources. Some entities often mentioned together and therefore one of them gets high values of cooccurrence features. For example, the baseline system answered the question *"when did tony romo got drafted?"* correctly, but since `Tony Romo` is often followed by `Dallas Cowboys`, Text2KB ranked the team name higher. Another common problem with our features is an artifact of entity linking, which works better for names and often skips abstract entities, like professions. For example, the correct answer to the question *"what did jesse owens won?"* is an entity with the name `Associated Press Male Athlete of the Year`, which is rarely mentioned or it's hard to find such mentions. Some problems were introduced by a combination of components. For example, for *"where buddha come from?"* a topical entity `Buddhism` was introduced from search results, and it generated `Gautama Buddha` as one of the answer candidates. This answer was ranked the highest due to large number of mentions in the search results.

In summary, we show that ideas behind Text2KB could be integrated into other systems and improve their performance. The error analysis suggested that even though a significant number of questions in the WebQuestions dataset have incorrect or ambiguous ground truth labels, there is still a room for improvement. In particular, the future work for Text2KB will include a better strategy for entity linking using external data sources and a better context model for entity mentions in text documents, which can put more weight on entities mentioned in the context related to the question.

6. RELATED WORK

One well known annual benchmark in knowledge base question answering is QALD (Question Answering over Linked Data), started in 2011 [23]. These benchmarks use the DBpedia knowledge base and usually provide a training set of questions, annotated with the ground truth SPARQL queries. In QALD-3 a multilingual task has been introduced, and since QALD-4 the hybrid task is included. This task asks participants to use both structured data and free form text available in DBpedia abstracts. The formulation of the hybrid task is the most relevant to our work, but there are some key differences. Questions in the hybrid track are manually created in such a way, that they can *only* be answered using a combination of RDF and free text data. Secondly, the hybrid task focuses on text data already present in a KB, whereas we are exploring external text resources. In general, because of the expensive labeling process, QALD datasets are relatively small, for example, QALD-5 training set for multilingual question answering includes 300 examples and 40 for the hybrid task, with 50 and 10 test questions correspondingly. Therefore, due to the scale of datasets and slightly different focus of tasks, we did not evaluate our techniques on the QALD benchmarks, but intend to explore it in the future.

Another benchmark dataset – WebQuestions – was introduced by Berant et al. [6]. The approaches proposed since then differ in the algorithms used for various components, and, what is more relevant to our work, the use of external datasets. WikiAnswers corpus of question clusters can be used to learn a question paraphrasing model, which helps to account for different ways a question can be formulated [7]. Another approach to learn term-predicate mappings is to mine them from a large text corpus [30], weakly labeled using distant supervision [18]. In the current paper, we build on this idea in two ways: by introducing a new data source (CQA archives), and by mining a language model for each mentioned entity pair, rather than predicates. Another approach to generate more training data is to automatically convert RDF triples to questions using entity and predicate names [10]. Finally, many systems work with distributed vector representations for words and RDF triples and use various deep learning techniques for answer selection [10, 31]. In all of these works, external resources are used to train a lexicon for matching questions to particular KB queries. In our work, we use external resources in a different way: we are targeting better candidate generation and ranking by considering the actual answer entities rather than predicates used to extract them.

In general, combining different data sources, such as text documents and knowledge bases, for question answering has been attempted before, and it has been already implemented in hybrid QA systems [2, 5]. Such systems typically have different pipelines that generate answer candidates from each of the data sources independently, and merge them to select the final answer at the end. We make a step towards integration of approaches, by incorporating text resources into the different stages of knowledge base question answering process. This is similar to the work of [22], who explored the use of entity types and descriptions from a KB for text-based question answering, and [13] explored such semantic annotations for ad-hoc document retrieval.

An alternative approach to QA is by using Open Information Extraction [15], which extract semi-structured data from text. OpenIE repositories can be queried using structured query languages, and at the same time allows keyword matching against entities and predicates [16]. In this work, we are borrowing an idea of learning about entity relationship via natural language phrases connecting them. However, since we do not need to extract clean set of relation tuples, we can keep all kinds of phrases, mentioned around entity pairs.

M. Yahya et al [27] proposed extending SPARQL triple patterns with text keywords, and using certain query relaxation techniques to improve the robustness of KBQA systems. Query relaxation drops certain triple patterns from SPARQL query and adds the corresponding question words as keywords to other triple patterns. The idea of query relaxations and using text in SPARQL queries was extended in [26], which proposed a framework for querying extended knowledge graphs, comprising of a combination of KB and OpenIE triples. These ideas are complimentary to our work, because our use of text data improves the matching between question phrases and KB concepts, whereas query relaxations are applied when a good match wasn't found. Another KB-Text hybrid approach, proposed in [25], utilizes text resources as a post-processing step for answer validation and filtering. In contrast, Text2KB integrates external textual information into all stages of question answering, resulting in more robust and overall higher performance than previously explored enhancements done in isolation.

7. CONCLUSIONS AND FUTURE WORK

Our work showed that unstructured text resources can be effectively utilized for knowledge base question answering to improve query understanding, candidate answer generation and ranking. We focused on three particular techniques and associated text information sources: web search results for query understanding and candidate ranking, community question answering data for candidate generation, and text fragments around entity pair mentions for ranking. Certainly, there are more resources that could be potential adapted, *e.g.,* entity profile pages like Wikipedia, news sources, textbooks, and many others. However, we believe that the proposed approach is general enough that it could be extended and successfully incorporate these other diverse text sources.

In the future, we plan to extend our work to the more open setup, similar to the QALD hybrid task, where questions no longer have to be answered exclusively from the KB. This would require extending the described techniques, and creating new QA benchmarks.

ACKNOWLEDGMENTS: This work was partially supported by the Yahoo Labs Faculty Research Engagement Program (FREP). We also thank the anonymous reviewers for their constructive comments.

8. REFERENCES

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. *Dbpedia: A nucleus for a web of open data*. Springer, 2007.

[2] K. Barker. Combining structured and unstructured knowledge sources for question answering in watson. In *DILS*, Lecture Notes in Computer Science. Springer, 2012.

[3] H. Bast and E. Haussmann. More accurate question answering on freebase. Proceedings of CIKM, 2015.

[4] P. Baudiš. Systems and approaches for question answering. 2015.

[5] P. Baudiš and J. Šedivỳ. Modeling of the question answering task in the yodaqa system. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*. Springer, 2015.

[6] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*, 2013.

[7] J. Berant and P. Liang. Semantic parsing via paraphrasing. In *Proceedings of ACL*, 2014.

[8] J. Berant and P. Liang. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3, 2015.

[9] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of ICMD*, 2008.

[10] A. Bordes, S. Chopra, and J. Weston. Question answering with subgraph embeddings. In *Proceedings of EMNLP*, 2014.

[11] E. Brill, S. Dumais, and M. Banko. An analysis of the askmsr question-answering system. In *Proceedings of EMNLP*. Association for Computational Linguistics, 2002.

[12] M. Cornolti, P. Ferragina, M. Ciaramita, H. Schütze, and S. Rüd. The smaph system for query entity recognition and disambiguation. In *Proceedings of the First International Workshop on Entity Recognition and Disambiguation*, 2014.

[13] J. Dalton. *Entity-based Enrichment for Information Extraction and Retrieval*. PhD thesis, University of Massachusetts Amherst, 2014.

[14] H. T. Dang, D. Kelly, and J. J. Lin. Overview of the trec 2007 question answering track. In *Proceedings of TREC*, 2007.

[15] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proceedings of EMNLP*, 2011.

[16] A. Fader, L. Zettlemoyer, and O. Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of SIGKDD*, 2014.

[17] J. Lin. An exploration of the principles underlying redundancy-based factoid question answering. *Transactions of ACM*, 25(2), Apr. 2007.

[18] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL*, 2009.

[19] J. Pound, P. Mika, and H. Zaragoza. Ad-hoc object retrieval in the web of data. In *Proceedings of WWW*, 2010.

[20] D. Savenkov, W.-L. Lu, J. Dalton, and E. Agichtein. Relation extraction from community generated question-answer pairs. In *Proceedings of NAACL: Student Research Workshop*, 2015.

[21] V. I. Spitkovsky and A. X. Chang. A cross-lingual dictionary for english wikipedia concepts. In *Proceedings of LREC*, 2012.

[22] H. Sun, H. Ma, W.-t. Yih, C.-T. Tsai, J. Liu, and M.-W. Chang. Open domain question answering via semantic enrichment. Proceedings of WWW, 2015.

[23] C. Unger, C. Forascu, V. Lopez, A.-C. N. Ngomo, E. Cabrio, P. Cimiano, and S. Walter. Question answering over linked data (qald-5). In *Proceedings of CLEF*, 2015.

[24] D. Vrandečić and M. Krötzsch. Wikidata: A free collaborative knowledgebase. *Communications of ACM*, (10), Sept. 2014.

[25] K. Xu, Y. Feng, S. Reddy, S. Huang, and D. Zhao. Enhancing freebase question answering using textual evidence. *arXiv preprint arXiv:1603.00957*, 2016.

[26] M. Yahya, D. Barbosa, K. Berberich, Q. Wang, and G. Weikum. Relationship queries on extended knowledge graphs. In *Proceedings of WSDM*, 2016.

[27] M. Yahya, K. Berberich, S. Elbassuoni, and G. Weikum. Robust question answering over the web of linked data. In *Proceedings of the CIKM*, 2013.

[28] X. Yao. Lean question answering over freebase from scratch. In *Proceedings of NAACL Demo*, 2015.

[29] X. Yao, J. Berant, and B. Van Durme. Freebase qa: Information extraction or semantic parsing? In *Proceedings of ACL*, 2014.

[30] X. Yao and B. Van Durme. Information extraction over structured data: Question answering with freebase. In *Proceedings of ACL*, 2014.

[31] W.-t. Yih, M.-W. Chang, X. He, and J. Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of ACL*, 2015.