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

ABSTRACT

One of the major challenges for knowledge base question answering systems (KBQA) is to translate a natural language question to 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 enrich the available information. We introduce a new system, Text2KB, that connects a KB with external text, specifically, we revisit different phases in the KBQA process and demonstrate that text resources improve question interpretation, candidate generation, filtering and ranking. Starting with the best publicly available system, Text2KB utilizes web search results, community question answering and text document collection data, to detect question topic entities and enrich the features of the candidates derived from the KB. Text2KB significantly improves on the initial KBQA system, and reaches the best known state of the art performance on a popular WebQuestions knowledge base question answering dataset. The results and insights developed in this work are both practically useful, and 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, factual, or factoid questions, have been an active focus of research for decades due to both practical importance and relatively objective evaluation criteria. As a particularly important example, a large proportion of Web search queries are looking for entities or their attributes (**CITATION**), a setting on which we focus in this work.

Two relatively separate approaches Question Answering (QA) have emerged: text-centric, or Text-QA and knowledge-base-centric, or KB-QA. In the more traditional, text-QA approach, QA systems used text document collections to

SIGIR '16 Pisa, Italy

© 2016 ACM. ISBN 123-4567-24-567/08/06...$15.00

DOI: 10.475/123_4

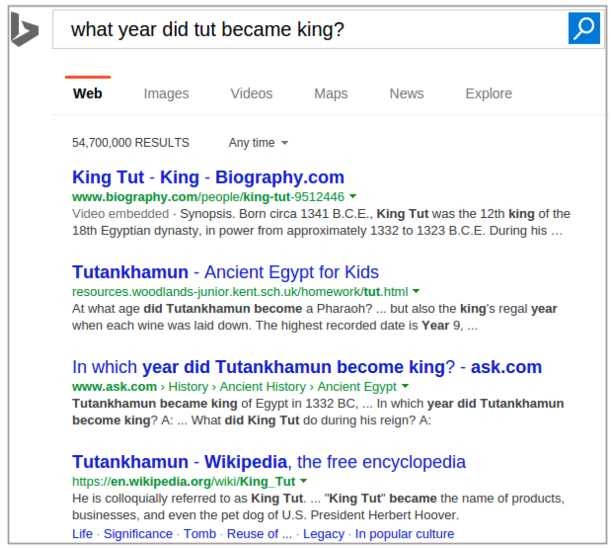

Figure 2: Search results for the question "what year did tut became king?"

retrieve passages relevant to a question and to extract candidate answers [21]. Unfortunately, an unstructured text passage does not provide explicit information about the candidate entities, and has to be inferred from the question text. The KB-QA approach, which evolved from the database community, relies on large scale knowledge bases, such as dbPedia [1], Freebase [8] and WikiData [21], 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 need to eventually deal with natural language questions, in which information needs are expressed by the vast majority of users. While question understanding is difficult in itself, this setting is particularly challenging for KB-QA systems, as it requires a translation of a text question into a structured query language. That is challenging for a number of reasons, including the complexity of a KB schema, and many differences between natural language and knowledge representations. For example, Figure 1 gives a SPARQL query that retrieves the answer to a relatively simple question *"who was the president of the Dominican Republic in 2010?"* from Freebase.

Any KB-QA systems must address three challenges, namely

```
PREFIX : <http://rdf.freebase.com/ns/>
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 that retrieves the answer to the query *"who is the current president of the dominican republic in 2010?"*

question entity identification to anchor the query process; candidate answer entity identification; and candidate ranking. We will show that these challenges can be alleviated by the appropriate use of external textual data.

The first problem that a KBQA system faces is question entity identification. The performance of the whole system greatly depends on this stage [22], because it seeds the answer candidate search process. Question text is often quite short, may contain typos and other problems, that complicate question entity identification. Existing approaches are usually based on dictionaries that contain entity names, aliases and some other phrases, which were used to refer to entities [18]. These dictionaries are often 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"*. A mapping *tut* → *"Tutankhamun"* is missing in the dictionary used by one of the state of the art systems and therefore it couldn't answer this question correctly. Instead of increasing the dictionary size we propose to use web search results to find variations of question entity names, which can be easier to link to a KB. This idea has been shown effective in entity linking in web search queries [11], one of the tracks on the Entity Recognition and Disambiguation Challenge 2014[1]. Figure 2 presents web search results for the query *"what year did tut became king?"*, which shows that indeed many documents mention the full name of the entity, which in turn can be more easily mapped to a KB entity.

After question entities have been identified, the second challenge is exploring their neighborhoods in the KB to generate candidate answers. A query addresses one or multiple KB predicates, which should be somehow related to words and phrases in the question, and somehow ordered in order to select the best answer. Existing knowledge base question answering approaches [3, 5, 6, 7, 9, 23] rely on some kind of a lexicon, which is learned from manually labeled training data, and supported by additional resources, such as question paraphrases [6] and weakly labeled sentences from a large text collection [24]. However, since manually labeled training data tends to be limited, such lexicons do not cover thousands of different predicates often present in a KB. 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* in the training set. For example, an RDF triple `[Bigos, food.dish.type_of_dish1, Stew]` answers a test question *"what are bigos?"*, but there are no questions from the training set that are answered using the same predicate. In addition, even if the training set contained an example targeting a particular KB predicate, the lexicon might not cover all the other possible ways the same information can be asked about. For example, a test question in the WebQuestions dataset is *"who is the woman that john edwards had an affair with?"*. This question is similar to, and is answered with a similar query as a training set question *"who did jon gosselin cheat with?"*, but the word *affair* isn't used in the training set. On the other hand, traditional text-based question answering systems benefit from the redundancy of information on the Web, where the same information is stated in many different ways in many documents [15]. 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 [10]. Thus, to address this challenge, our work adapts ideas from text-based question answering to enrich the representation of candidate structured queries with additional text documents and fragments, that can help to select the best answer. For example, 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. Finally, the third challenge of a KBQA system is how to select the best candidate answer among many candidates. We show that enriching the candidate answers with features derived from external text results, significantly improves the ranking.

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 natural language resources connected via entity links. 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 knowledge base question answering, via entity linking. We introduce three techniques: enhancing question entity identifi-

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

cation by analyzing 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 using 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 a number of techniques of using external text that significantly improve the performance of the KBQA approach. More broadly, our work bridges the gap between Text-QA and KB-QA worlds, demonstrating an important step forward towards combining unstructured and structured data for question answering.

2. INFORMATION EXTRACTION APPROACH TO KBQA

Traditionally, people consider semantic parsing and information extraction approaches to knowledge base question answering [23]. The former focuses on question understanding and attempts to parse the sentences into some kind of semantic representation, *e.g.,* logical forms [5, 6, 7]. Information extracting approaches [3, 25, 24] are based on identifying question topical entities, exploring the neighborhood of these entities in a KB using a set of query template and ranking these candidates. Theoretically, semantic parsing approaches are capable of generating complex queries, which are hard to cover with a reasonable set of templates. But in practice, answers to most of the questions lie within two edges in a KB, and that is why information extraction approaches turn out to be very effective.

One of the most popular benchmark datasets for knowledge base question answering is WebQuestions [5], which has attracted a lot of attention recently and as a result the performance increased from 0.357 [5] to 0.525 [25] in average F1 over test questions. The dataset is based on Freebase, which has been recently shut down[2]. The shutdown of Freebase means that it will no longer be extended, but all the data will be available and it will be merged with WikiData[3]. Therefore, future datasets should probably use different reference KB, but there is no problem in using Freebase for experiments on existing benchmarks, such as WebQuestions. We should also stress, that the proposed approach isn't tied to Freebase and can be applied for question answering over dbPedia, WikiData or other knowledge bases.

The focus of this work is on the fusion between structured data in the KB and unstructured text data. Therefore, we chose to extend an existing KBQA system: Accu [3]. It follows an information extraction approach to KBQA and achieves one of the highest scores among publicly available systems. However, our approach can be incorporated into other systems as well.

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

2.1 Our baseline system

Let's consider a question *"who is the woman that john edwards had an affair with?"*. First, the system identifies a set of possible question entities. In our example, entity `John Edwards` with Freebase mid `/m/01651q` is the main question entity. However, Freebase contains millions of entities and it's often hard 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`), `John Edwards` an American sports car racing driver (`/m/06zs089`) and other people with the same name. There is even an entity with the name "`had an affair with`"[4]. Accu considers all spans of terms under certain conditions on POS tags and use a dictionary of names, aliases and anchor tests [18] to map phrases to potential entities. Most recent systems, including Accu, don't disambiguate entities at this stage and keep a set of candidates along with some information about entity popularity and mention scores.

On the next stage answer SPARQL query candidates are generated by exploring the neighborhood of the question topical entities usign a predefined set of query templates. Each query template has an entity, predicate and answer entity placeholders. Majority of the answers in 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 multi-argument relations also mentions another question entity, *e.g.,* `Captain Kirk` and `Star Trek` for the question *"who played captain kirk in star trek movie?"*.

Finally, each query candidate is represented with a set of features, that include information about the popularity of question entities (mention frequency), entity linking mention score, size of the answer entity list, how many tokens

[2]https://goo.gl/SZC3tg
[3]https://www.wikidata.org/
[4]http://www.freebase.com/m/0c0n01x

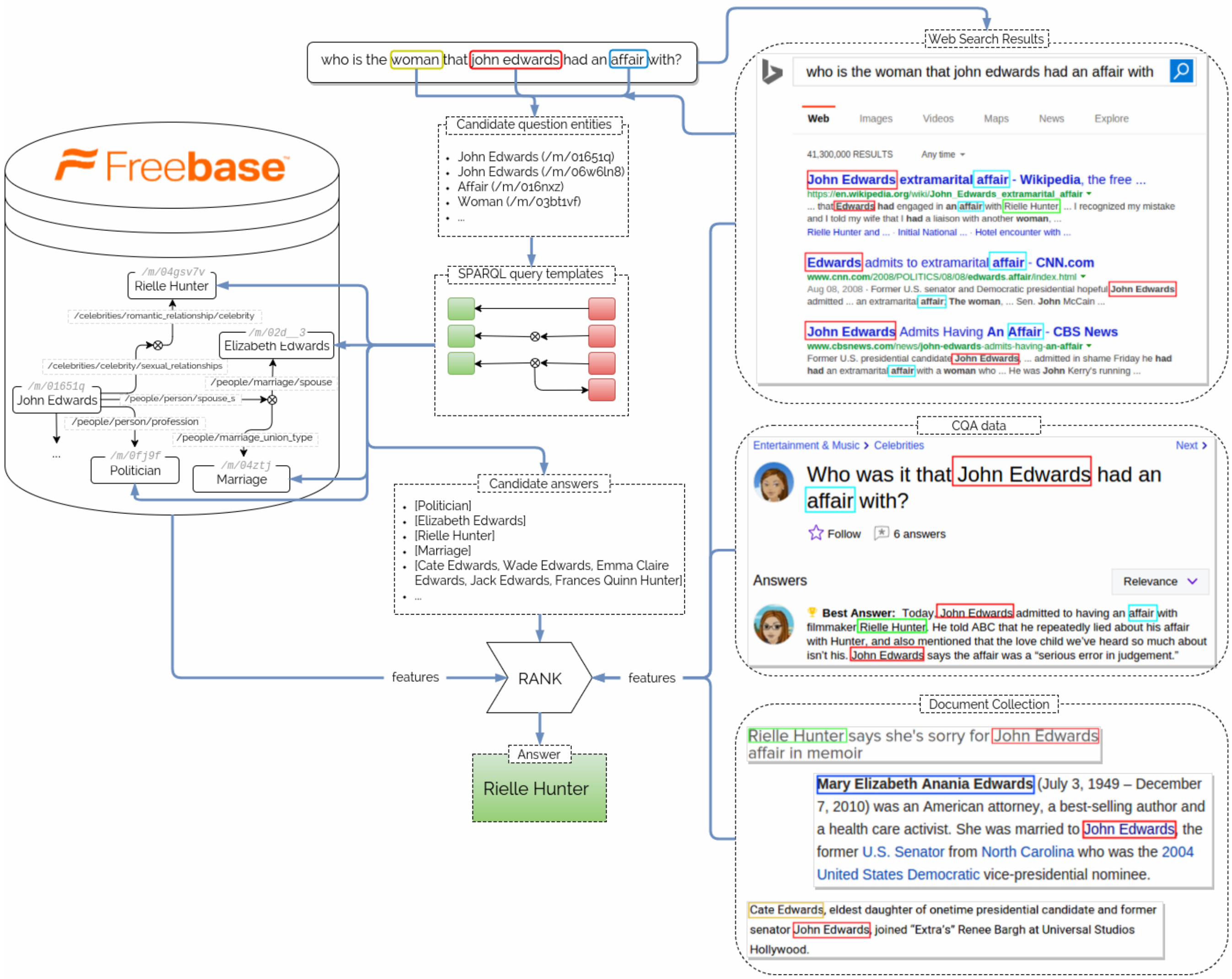

Figure 3: The architecture of our Text2KB Question Answering system

from the questions match to entities used in the candidate, how many tokens match predicates via exact match by its name or using a dictionary of terms learned for each predicate using distant supervision from a large text corpus, *e.g.,* ClueWeb[5], and some other. One of the most useful features is a score from a logistic regression model, that predicts whether a predicate answers the question represented as a set of uni- and bigrams. The full list of features can be found in the original paper [3].

The final stage of the question answering process is query candidate filtering and ranking. To rank candidates are sorted in such a way, that pairs of candidates are compared using trained random forest model.

2.2 Baseline system extensions

Preliminary analysis of the system suggested a couple of directions for improvement, that do not require an external data sources. First, we noticed that since the system doesn't really use information about answer entity Freebase types, in many cases it returns an answer that is type incompatible with the question: *e.g.,* state instead of county *etc.* Similarly to how relations are scored in Accu, we decided to train a model to predict how likely a certain notable entity type is the answer of a question, represented as a set of uni- and bigrams. The score of this model is used as a feature for candidate ranking.

A second introduces a new date range query template.

```
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
   )
}
```

This template helps to solve the cases like *"what team did*

[5]http://www.lemurproject.org/clueweb12/

david beckham play for in 2011?", where we need to look at the ranges of dates of figure out in which range does the specified date falls. We also experimented with additional template, that filters out lists by entity notable types.

3. TEXT2KB: INCORPORATING TEXT DATA INTO KBQA

The architecture of our system, called Text2KB, is presented on Figure 3. The left part of the picture roughly corresponds to the architecture of existing information extraction approaches to knowledge base question answering. The right part introduces additional different external text data sources, which are integrated into the question answering pipeline on multiple stages. In this work we use web search results, community question answering (CQA) data and we use a large collection of documents with detected KB entity mentions. But besides external text data, many knowledge bases including Freebase contain some text data as well. In Freebase most of the entities contain a description paragraph, which often comes from the entity Wikipedia profile. These descriptions of entities in the KB were found useful for text-based question answering [19]. For completeness, we decided to include them in our system as well. Each description is represented as a vector of tokens and as a vector of mentioned entities and we compute a cosine similarity between the question tokens and entity vectors and use these scores as features for candidate ranking. In a similar we can incorporate any other entity profile text, such as full Wikipedia article.

3.1 Web search results

We start by issuing a question as a query to a commercial web search engine[6] and extract top 10 search result snippets and get the corresponding documents. Document snippets are usually built to present the information most relevant to the query and often contain answers to a question. Unfortunately, for longer queries the snippets often represent and combination of small phrases, that contain mostly question terms and very few additional information. Nevertheless, we keep both snippets and documents text, and using system's linker detect KB entity mentions. This data turns out to be useful for multiple purposes, *i.e.,* question entity identification and answer candidate ranking.

Question entity identification. Question text provides a very limited context for entity disambiguation and in addition the entity name can be misspelled or an uncommon variation can be used. This complicates question topical entity identification, which is the foundation of whole question answering process. Luckily, web search results help with these problems as they usually contain multiple various mentions of the same entities and provide more context for disambiguation.

To extend the set of detected question entities Text2KB uses search results snippets. There are multiple mentions of different entities, to filter out only entities, that are also mentioned in the question we use string distance. More specifically, we take names of all entities detected in the question and compute their term by term similarity with non-stopwords from the question. In this work we used Jaro-Winkler string distance and and entity was added to the list of question entities if at least one of its tokens e_t have high similarity with one of the question tokens q_t excluding stopwords ($Stop$), *i.e.,* :

$$\max_{\substack{e_t \in M \backslash Stop \\ q_t \in Q \backslash Stop}} distance_{Jaro-Winkler}(e_t, q_t) \geq 0.8$$

Answer candidate features. Most of the information stored in knowledge bases is also present in other formats, including natural language statements, tables, *etc.* For example, on Figure 2 multiple snippets mention the date when Tutankhamun became the king. Text-based question answering system usually generate answer candidates from passages extracted from retrieved documents. In our case candidates are already generated in a form of KB queries, that return certain subsets of entities. Text2KB uses snippets and documents to compute a set of features, which are used for answer candidate ranking. More specifically we do this following:

1. Precompute term and entity IDFs[7]. We used Google n-grams corpus to approximate terms IDF by collection frequency and available ClueWeb Freebase entity annotations[8] to compute entity IDFs
2. Each snippet and document is represented by two TF-IDF vectors of lowercased tokens and mentioned entities
3. In addition, vectors of all snippets and all documents are merged together to form combined token and entity vectors
4. Each answer candidate is also represented as TF-IDF vectors of terms (from entity names) and entities
5. We compute cosine similarities between answer and each snippet and document token and entity vectors. This gives us 10 similarity scores for every document for token vectors and 10 similarities for entity vectors, we take average and maximum scores as features.
6. We do the same for the combined document and use cosine similarities as features

3.2 Community Question Answering data

Manual labeling of questions with answers is expensive and therefore largest KBQA datasets still contain just several thousands of examples, which is too small for data intensive approaches to learn good strategies for mapping questions into millions of KB entities and predicates. Researchers have proposed to use weakly supervised methods to extend the lexicon with mappings learned from statements mentioning entity pairs from a large corpus [24]. However, often there exist a lexical gap between how information is asked about in a question and how it is expressed in a statement. On the other hand there are huge archives of questions and answers posted by real users on various community question answering websites, *e.g.,* Figure 4.

For our experiments we took 4,483,032 questions from Yahoo! Comprehensive Questions and Answers WebScope dataset[9]. Texts of each question and answer pair were run through an entity linker, that detected mentions of Freebase entities. Next, similar to an idea of relation extraction from CQA data [17], we use distant supervision to label each

[6]https://datamarket.azure.com/dataset/bing/search

[7]https://en.wikipedia.org/wiki/Tf-idf

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

[9]https://webscope.sandbox.yahoo.com/catalog.php?datatype=l

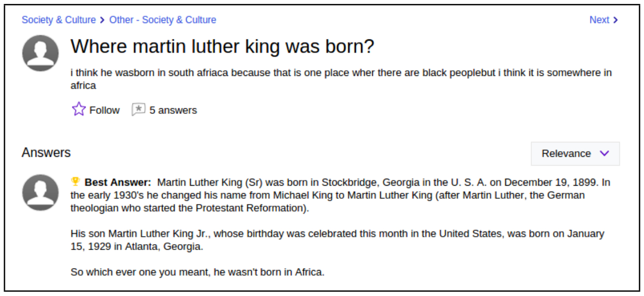

Figure 4: Example of a question and answer pair from Yahoo! Answers

Table 1: Examples of term-predicate pairs with high PMI scores

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
illness	medicine.symptom.symptom_of	2.11
	medicine.decease.causes	1.68
	medicine.disease.treatments	1.59
win	sports.sports_team.championships	4.11
	sports.sports_league.championship	3.79

question-answer pair with relations between entities mentioned in the question and in the answer. As a result we have a set of questions, annotated with KB predicates connecting some question and answer entities, which are, often incorrectly, assumed to answer the question. We learn associations between question terms and predicates by computing pointwise mutual information scores[10] (PMI) for each term-predicate pair. Examples of scores for some terms from WebQuestions dataset questions are given in Table 1.

Although noisy, the statistics look reasonable to be used for candidate ranking. In Text2KB we take candidate answer predicates and lookup PMI scores between these predicates and terms in the question. Missing pairs are given a score of 0, and minimum, average and maximum of these scores are used as features. Since this kind of data is usually sparse, we decided to consider vector space embeddings of terms to solve a problem of synonym terms with missing score. We use pretrained word2vec word embeddings[11] to generate predicate embeddings by taking weighted average of term vectors from predicate's PMI table. Each term's embedding vector is weighted by its PMI value (terms with negative score are skipped). Then, we compute cosine similarities between predicate vector and question term vectors and take their minimum, average, maximum as features. Similarity between the predicate vector and average question term vector is also computed.

[10]https://en.wikipedia.org/wiki/Pointwise_mutual_information
[11]https://code.google.com/p/word2vec/

Table 2: Examples of entity pairs language model data

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

3.3 Entity pair language model

When ranking candidate answer, we are interested in estimating if topic and answer entities are related in a way asked in the question. Existing systems usually look on how candidate predicates are expressed in questions and statements. But predicate isn't the only way we can look at this, another alternative is to consider text pieces, *e.g.,* sentences, that mention topical and answer entities together. For example, in the bottom right corner of Figure 3 we can see some passages that mentioned a pair of people, and the context of these mentions often expresses the nature of the relationships between the entities. This resembles OpenQA approach of [14] with a difference, that information isn't filtered out by keeping only some sentences and converting them to a triple format. Moreover, since these relationships are now expressed in a natural language, we can consider various measures of text similarity between them and the question.

We take ClueWeb12 corpus with existing Freebase entity annotations[12] and compute counts of different terms that occur in the context to an entity pair mention. By an entity pair mention we mean a pair of mentions of different entities within 200 characters of each other. We take terms in between mentions and 100 character before and after mentions as the context. A small sample of this data is presented in Table 2.

First, given a set of question terms Q, an answer candidate, that includes question entity e_1, we compute a language model score for every answer entity e_2:

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

and use minimum, average and maximum as features. Similar to the way we handled CQA-based data, we use embeddings to handle the sparsity problem. For each entity pair a weighted (by counts) average embedding vector of terms is computed and again minimum, average and maximum cosine similarities between this vectors and question tokens vector is used as features.

4. EVALUATION

We followed the standard evaluation procedure for the WebQuestions dataset and used the original 70-30% train-

[12]http://lemurproject.org/clueweb12/FACC1/

test split, which results in 3,778 training and 2,032 test questions. Since each answer is potentially a list of entities a^*, the quality of an answer a is represented by F1-score:

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

where $precision(a^*, a) = \frac{|a^* \cap a|}{|a|}$ and $recall(a^*, a) = \frac{|a^* \cap a|}{|a^*|}$.

We also report average precision and recall, as well as an F1 score of average precision and recall. The results of existing approaches, our baseline and Text2KB systems is presented in Table 3.

As we can see, Text2KB significantly improves over the baseline system and reaches the current best published result - STAGG [25], and we believe that this system will also benefit from the ideas of our work, and we will explore this question in Section 5.

4.1 Ablation Study

To study effects of different components in isolation we made a series of ablation studies. For convenience, we introduce the following notations for different components introduced in our system:

- T - notable type score model as a ranking feature
- DF - date range filter-based query template
- E - using web search result snippets for question entity identification
- W - using web 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 `+<component>` to for a system with a certain component added, and `-<component>` when the component is removed. For example, the baseline system will be denoted as "`Accu`" according the authors notation. The same system with additional date range filter query templates and notable types score model is denoted as "`Accu +DF+T`", which represents the same system as "`Text2KB -E-W-CQA-CL`". Our full system "`Text2KB`" can be also denoted as "`Accu +DF+T+E+W+CQA+CL`".

The first question that we are asking is what are the improvements, introduced by adding date range filter templates, notable type model, entity linking from web search results and text-based features generated from all the different sources. Results of this ablation experiment are presented in Table 4. As we can see, additional date range filters and notable types model (`Text2KB -E-W-CQA-CL`) are responsible for an increased recall and a drop in precision compared to the baseline model. Detecting question entities (`Text2KB -W-CQA-CL`) help improve both precision and recall, and therefore average F1 score by 0.096 points. An even bigger improvement is achieved by introducing all our external text-based features, and since these improvements are independent, their combination boosts the performance even more.

Now, let's look into the relative importance of each of the data sources, we will remove or use a group of web search, cqa or clueweb-based features and see how the performance of the whole system change. Table 5 summarizes the results of these experiments.

Features that we generate from web search results are the most effective, because even without other data sources the QA performance is almost as high as the full system. In addition, if we remove web search results based features the performance drops more than for other text data sources. With CQA and ClueWeb based features the results are not that straightforward. Even though if used alone CQA-based features give us the lowest average F1 score among all the data sources, without these features the quality decreases. Whereas, removing ClueWeb-based features didn't cause a drop of the performance.

Table 4: Evaluation results for the baseline system and various subsystems introduced in Text2KB

System	avg R	avg Pr	avg F1
`Accu` (baseline)	0.604	0.498	0.494
`Text2KB -E-W-CQA-CL= =Accu +DF+T`	0.6169	0.4807	0.4987
`Text2KB -W-CQA-CL`	0.6272	0.4920	0.5083
`Text2KB -E`	0.6344	0.4966	0.5140
`Text2KB`	0.6354	0.5059	0.5223

Table 5: Evaluation study for our system with different text-based data sources used to generate features

System	avg R	avg Pr	avg F1
`Text2KB -W`	0.6327	0.4960	0.5126
`Text2KB -CQA`	0.6420	0.4987	0.5185
`Text2KB -CL`	0.6444	0.5047	0.5228
`Text2KB` (Web search results only)	0.6423	0.5028	0.5216
`Text2KB` (ClueWeb only)	0.6307	0.4978	0.5138
`Text2KB` (CQA only)	0.6224	0.4928	0.5077
`Text2KB`	0.6354	0.5059	0.5223

Since we used each data source to generate multiple different features for candidate ranking, it is interesting to see which particular features are more useful than others by the ranking machine learning algorithm (we used random forest). Figure 5 plots a subset of features ranked by their Gini index-based importance scores in the final answer candidate ranking model.

The figure supports the observation that web search results based features turned out to be the most useful among other text-based data sources. However, other text data sources also contribute to the overall improvement. According the model, best feature based on entity pair language model computed on ClueWeb dataset is more useful than CQA-based features.

5. ANALYSIS AND DISCUSSION

5.1 Generalization analysis

In this work 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.,* for example, can the result of the currently best system STAGG [25] be improved. The differences between our baseline system Accu and STAGG lie in the components, *i.e.,* entity linking algorithm, a set of query templates and ranking methods, therefore our approach is complementary and should be helpful.

Table 3: Performance of the Text2KB system on WebQuestions dataset

System	avg Recall	avg Precision	F1 of avg Prec and Recall	avg F1
SemPre [5]	0.413	0.480	0.444	0.357
Subgraph Embeddings [9]	-	-	0.432	0.392
ParaSemPre [6]	0.466	0.405	0.433	0.399
Jacana [24]	0.458	0.517	0.486	0.330
Kitt AI [22]	0.545	0.526	0.535	0.443
AgendaIL [7]	0.557	0.505	0.530	0.497
STAGG [25]	0.607	0.528	0.565	0.525
STAGG (no duplicates[13]) [25]	0.6067	0.5263	0.5634	0.5234
Accu (baseline) [3]	0.604	0.498	0.546	0.494
Our system: Text2KB	0.6354	0.5059	0.5633	0.5223

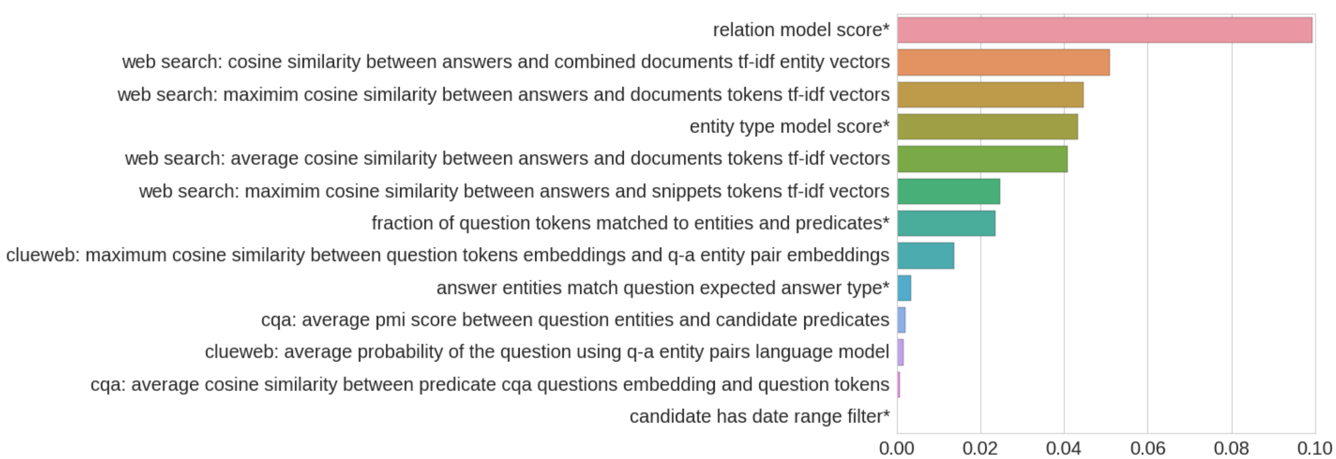

Figure 5: Importances of different text-based features for KBQA (features with * are not text-based and are provided for comparison)

Table 6: Results of experiments on combining Text2KB and STAGG predictions

System	avg Recall	avg Precision	avg F1
Accu (baseline) [3]	0.604	0.498	0.494
Our system: Text2KB	0.6354	0.5059	0.5223
STAGG [25]	0.607	0.528	0.525
Text2KB + STAGG	0.5976	0.5343	0.5320
Text2KB + STAGG (oracle)	0.7144	0.5904	0.6056

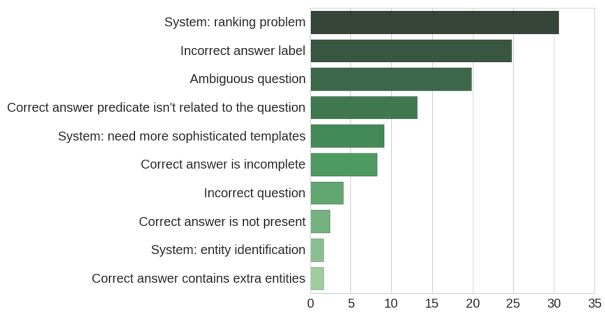

Figure 6: Distribution of problems with examples, where Text2KB returns an answer with F1≠1.0

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 slightly better average F1 score. Alternatively, we can look at the oracle combination of the systems, which always chooses an answer with higher F1. This experiment also verifies if both systems make the same mistakes, or there is a reasonable difference, and as we can see such a combination results in a performance of 0.6056, which is much higher than either of the systems.

WebQuestions dataset is rather small compared to the space of all possible KB queries a system should be able to generate and score. As a result, answers to 112 of the test questions involve a predicate that weren't observed in the training set, which may be a problem for approaches that rely on a lexicon, built during training. We evaluated both systems on these subset of questions, and indeed the performance turned out to be very low, *i.e.,* the average F1 score of Text2KB is 0.1640 compared to 0.1199 for STAGG. Unfortunately, the number of questions is too small to give statistically significant difference (p-value is 0.16 according to the paired t-test).

5.2 Error analysis

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

As we can see candidate ranking is still the major problem, however, it accounts for 31% of the cases. The second most popular problem is incorrect ground truth labels, which by our estimate accounted for almost a quarter of incorrect answers. For example: for the question *when tupac was shot?"* the label says `Tupac 1994 assault` instead of `Las Vegas` as the ground truth. A related set of questions have incomplete or overcomplete set of ground truth answer entities. A typical examples are questions asking for a list of movies, books, landmarks. The ground truth answer usually contains ~ 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. The profile page shows only a sample of 10 entities from large lists and the other are hidden behind the "NNN values total" link, which needs to be clicked in order to get the full list. 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 without any obvious preferences. For example, for the question *"where is shakira from?"* the ground truth is the country - `Colombia`, while Text2KB returned her place of birth - `Barranquilla`. Another example are questions like *"what did hayes do?"*, which 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 predicate related to the olympic sport competitions country participated in[14], which obviously isn't related to the question.

As for the system errors, we didn't observe any dominant problems. There are wins and loses introduced by each of our components. Web search results helped identify the right question topical entity in a number of cases, *e.g.,* the question *"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 provided more than enough evidence that romo refers to `Tomo 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.,* [11].

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 the `Charles Darwin` entity through the "known for" predicate[15] 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

[14] `olympics.olympic_participating_country.athletes`

[15] `user.lindenb.default_domain.scientist.known_for`

often mentioned together and therefore one of such entities 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 the question *"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 highest due to large number of mentions in the search results. In some cases search results aren't very relevant and only provide general information about question entity.

6. RELATED WORK

Recent development of large scale knowledge bases (e.g. dbPedia [1]) and Freebase [8]) motivated research in open domain question answering over linked data. As a result, in 2011 a series of QALD (Question Answering over Linked Data) evaluation campaigns has started. You can find the most recent report in [20]. These benchmarks use 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 was included. This task asks participants to build systems that can 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 a couple of 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, whereas WebQuestions dataset contains a more realistic set of questions, which doesn't require any 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 rather small, for example, QALD-5 training set for multilingual question answering includes 300 examples and 40 examples for the hybrid task. The evaluation was performed on 50 questions for multilingual task and just 10 for hybrid. Therefore, due to the scale of datasets and slightly different focus of tasks, we didn't attempt to evaluate our techniques on QALD benchmarks, but intend to explore it further in the future.

The benchmark, used in our work was introduced in [5]. The approaches proposed since are usually divided into semantic parsing [5, 6, 7] and information extraction [24, 22, 23, 25, 26] based approaches depending on whether the system build a semantic representation of the question utterance or just use string matching to rank answers. The systems differ in the algorithms used for different components, and, what is more relevant to our work, external datasets used. To account for different ways a question can be formulated [6] used a dataset of question clusters from WikiAnswers to learn a question paraphrasing model. Another approach to learn term-predicate mapping is to use distant supervision [16] to label a large text corpus, such as ClueWeb [24]. In this work we build on this idea and instead of focusing on term-predicate mappings, which might be too general, consider particular entity pairs. Freebase RDF triples can automatically converted to questions using entity and predicate names [9]. Finally, many systems work with distributed vector representations for words and RDF triples and use various deep learning techniques for answer selection [9, 25]. In all of these works, external resources are used to train a lexicon for matching questions to particular KB queries. The use of external resources in this work is different, 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 is not a novel idea, and it has been already implemented in hybrid QA systems [4, 2]. 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 different stages of knowledge base question answering process. This is similar to the work of [19], who explored the use of entity types and descriptions from a KB for text-based question answering. Such semantic enrichment of documents with KB entity annotations was also found effective for the ad-hoc retrieval problem [12].

We should also mention OpenIE [13], which represent an interesting mixture between text and structured data. Such knowledge repositories can be queried using query languages, *e.g.,* SPARQL, and at the same time allows keyword matching against entities and predicates. One can easily transform an existing KB to such a form by replacing predicates and entities with their names. This approach was losing to approaches based on a structured KB on WebQuestions, but had a better performance on a more general TREC QA and WikiAnswers datasets [14].

7. CONCLUSIONS AND FUTURE WORK

In this work showed that unstructured text resources can be effectively utilized for knowledge base question answering and improve query understanding, candidate generation and ranking. We focused our attention on three particular sources of text information: web search results, community question answering data and text fragments around entity pair mentions. Certaintly, there are more resources, that can be adopted, *e.g.,* entity profile pages like Wikipedia, Open IE knowledge bases, etc.

In the future, we plan to shift our attention to a more open setup, similar to the QALD hybrid task, where questions doesn't have to be answered completely from the KB. This will probably require a new dataset, which we intent to build.

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 O. Bodenreider and B. Rance, editors, *DILS*, volume 7348 of *Lecture Notes in Computer Science*, pages 53–55. Springer, 2012.

[3] H. Bast and E. Haussmann. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1431–1440, New York, NY, USA, 2015. ACM.

[4] P. Baudiš and J. Šedivỳ. Modeling of the question answering task in the yodaqa system. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 222–228. Springer, 2015.

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

[6] J. Berant and P. Liang. Semantic parsing via paraphrasing. In *Proceedings of ACL*, volume 7, page 92, 2014.

[7] J. Berant and P. Liang. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558, 2015.

[8] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA, 2008. ACM.

[9] A. Bordes, S. Chopra, and J. Weston. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pages 615–620, 2014.

[10] E. Brill, S. Dumais, and M. Banko. An analysis of the askmsr question-answering system. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 257–264. Association for Computational Linguistics, 2002.

[11] 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*, ERD '14, pages 25–30, New York, NY, USA, 2014. ACM.

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

[13] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics, 2011.

[14] A. Fader, L. Zettlemoyer, and O. Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1156–1165, New York, NY, USA, 2014. ACM.

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

[16] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009.

[17] D. Savenkov, W.-L. Lu, J. Dalton, and E. Agichtein. Relation extraction from community generated question-answer pairs. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 96–102, Denver, Colorado, June 2015. Association for Computational Linguistics.

[18] V. I. Spitkovsky and A. X. Chang. A cross-lingual dictionary for english wikipedia concepts. In N. C. C. Chair), K. Choukri, T. Declerck, M. U. DoÄ§an, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

[19] H. Sun, H. Ma, W.-t. Yih, C.-T. Tsai, J. Liu, and M.-W. Chang. Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1045–1055, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.

[20] 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 L. Cappellato, N. Ferro, G. J. F. Jones, and E. SanJuan, editors, *CLEF (Working Notes)*, volume 1391 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[21] D. Vrandečić and M. Krötzsch. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, Sept. 2014.

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

[23] X. Yao, J. Berant, and B. Van Durme. Freebase qa: Information extraction or semantic parsing? *ACL 2014*, page 82, 2014.

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

[25] W.-t. Yih, M.-W. Chang, X. He, and J. Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Association for Computational Linguistics (ACL)*, 2015.

[26] L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*, 2014.