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 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

Traditionally question answering systems used text document collections to retrieve passages relevant to a question and to extract candidate answers [21]. Unfortunately, a paragraph of text encodes a very limited amount of information about answer candidates and predictions has to be used instead, *e.g.* most of the systems estimate candidate answer entity type to match against the expected type inferred from the question text. On the other hand, modern large scale open-domain knowledge bases, such as dbPedia [1], Freebase [8] and WikiData [21] 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. Of course, regular users would rather prefer to ask natural language questions. Translation of text questions into structured query languages is very challenging for a number of reasons: complexity of a KB schema, variability of natural language and knowledge representation *etc.* For example, Figure 1 gives a SPARQL query that retrieves the answer to a relatively easy question *"who is the current president of the dominican republic in 2010?"* from Freebase.

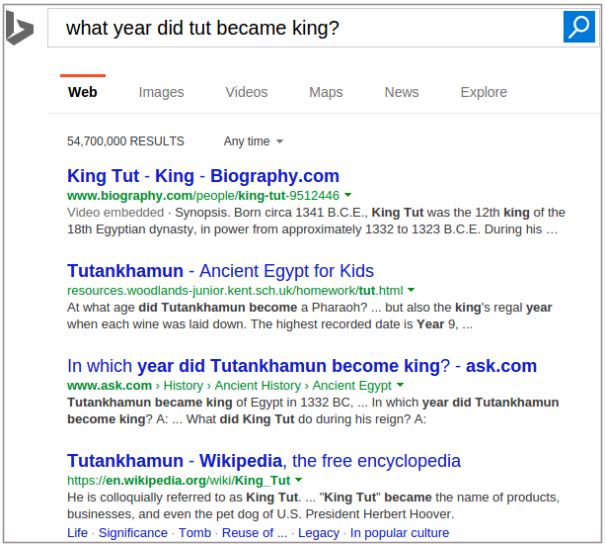

Figure 2: Search results for the question "what year did tut became king?"

The first problem that a KBQA system is facing 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

SIGIR '16 Pisa, Italy

DOI: 10.475/123_4

```
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?"*

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 was used for entity linking in web search queries [12], which was given as 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, which can easily be mapped to a KB entity.

After question entities have been identified the next step is to explore their neighborhood and build structured queries as candidate answers. A query addresses one or multiple KB predicates, which should be somehow related to words and phrases in the question and systems try score these mappings 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 some additional resources, such as question paraphrases [6] and weakly labeled sentences from a large text collection [24]. However, given the fact that manually labeled training data is very limited, such lexicons do not cover thousands of different predicates present in a KB. By our estimate in a popular WebQuestions KBQA dataset answers to ∼5.5% of test questions (112 out of 2032) involve a predicate, that doesn't 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 training set contains 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, test question *"who is the woman that john edwards had an affair with?"* is similar in meaning 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 with which 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]. We propose to borrow some ideas from text-based question answering and 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, community question answering page and text fragments mentioning pairs of entities, that can clearly be useful to answer the question about John Edwards' affair.

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

1.1 Contributions

Our main contributions in this work 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 for knowledge base question answering. Entity linking allows us to connect test resources with a KB. We explore three different types of text data, that is useful for KBQA: web search results (Section 3.1), Community Question Answering data (Section 3.2) and entity pairs language model based on a large text corpus (Section 3.3).

- evaluation and analysis. We evaluate the performance of our system on a popular WebQuestions dataset and demonstrate that using additional text resources we can improve the quality of knowledge base question answering (Section 4). In addition, we conduct an extensive analysis of the system and suggest directions for future research (Section 5).

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 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 {
```

[2]https://goo.gl/SZC3tg
[3]https://www.wikidata.org/
[4]http://www.freebase.com/m/0c0n01x
[5]http://www.lemurproject.org/clueweb12/

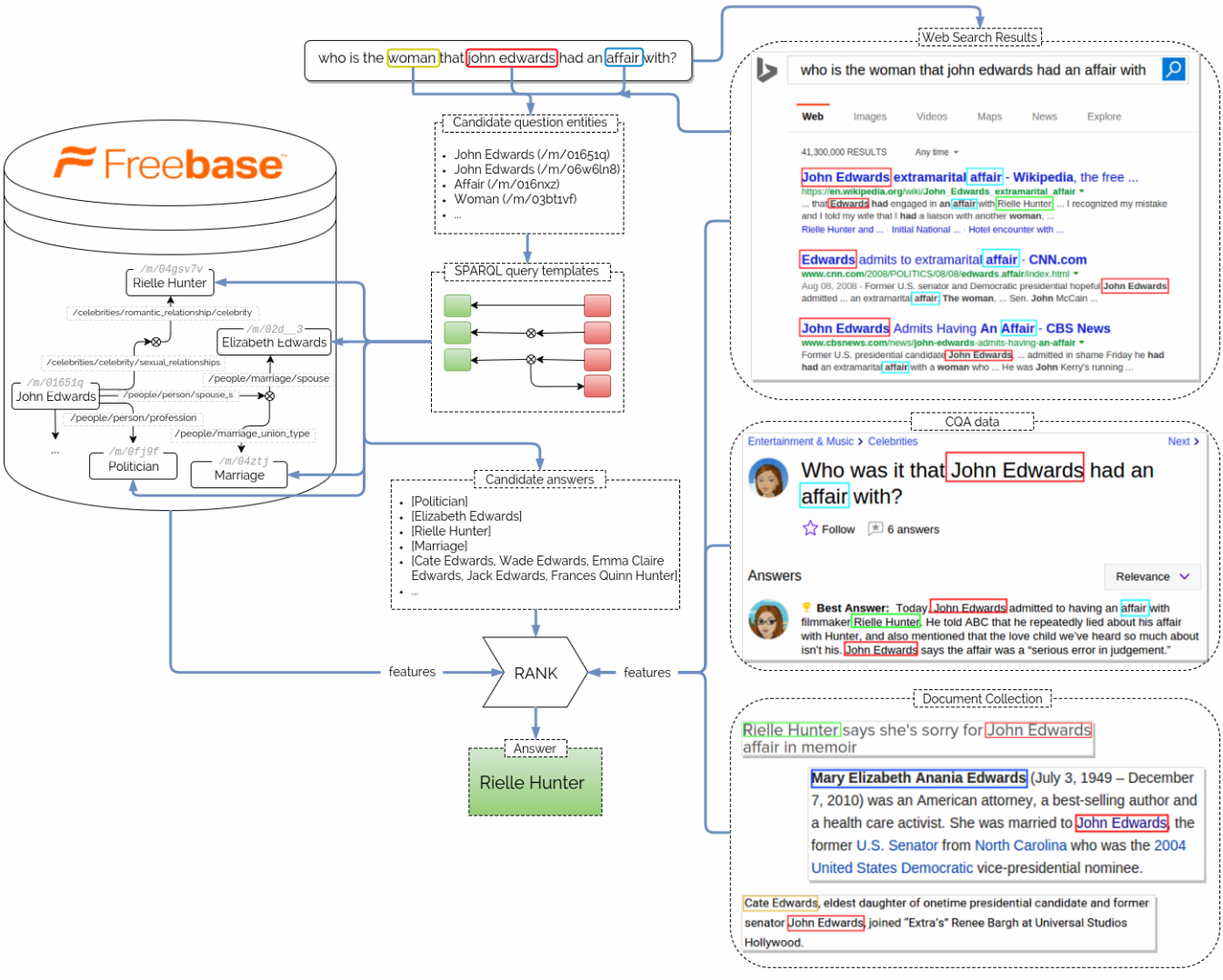

Figure 3: The architecture of our Text2KB Question Answering system

```
    <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 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

[6]https://datamarket.azure.com/dataset/bing/search
[7]https://en.wikipedia.org/wiki/Tf-idf
[8]http://lemurproject.org/clueweb09/FACC1/

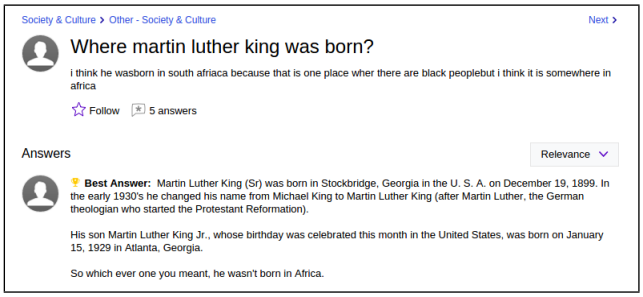

Figure 4: Example of a question and answer pair from Yahoo! Answers

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 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

[9]https://webscope.sandbox.yahoo.com/catalog.php?datatype=l
[10]https://en.wikipedia.org/wiki/Pointwise_mutual_information

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

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.

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.

[11]https://code.google.com/p/word2vec/

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

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...

4. EVALUATION

We followed the standard evaluation procedure for the WebQuestions dataset and used existing train-test splits for reporting our results. The benchmark uses average F1 score, which gives a partial credit to list answers that are not equal but overlap with the correct answer. 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]. However, we believe that this system will also benefit from the ideas of our work. To support this claim we combined results of STAGG and Text2KB systems using a simple heuristic: we prefer STAGG result when its answer length is shorter than Text2KB. This heuristic is inspired by the fact, that STAGG has a filtering stage, that filters out result lists based on some criteria, *i.e.* gender, entity type, *etc.* However, there is still a room for improvement if the combination is done inside a system, because the errors of STAGG and Text2KB differ, and if we use an oracle, which chooses an answer with the higher F1 score, we can achieve an average F1 of 0.6056.

We also evaluated both STAGG and Text2KB on 112 questions, where the correct answer is retrieved via a predicate, that weren't used in the positive training examples. These questions are harder to answer, and the average F1 score of Text2KB is 0.1640 versus 0.1199 for STAGG. This gives some anecdotal evidence, that by incorporating external text resources we were able to answer these questions slightly better, how these difference isn't statistically significant according to paired t-test (p-value 0.16).

4.1 Ablation Study

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
Text2KB + STAGG	0.5976	0.5343	0.5641	0.5320
Text2KB + STAGG (oracle)	0.7144	0.5904	0.6465	0.6056

To study individual effects of different components we made an ablation study. 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 result snippets and documents to generate features for candidate ranking
- CQA - using `[question term, KB predicate]` PMI scores, computed from CQA collection of question-answer pairs, to generate features for candidate ranking
- CW - using entity pairs language model, computed on a large text collection, to generate features for candidates ranking

In our results table we will use the notation `+<component>` to denote 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 features based on external free text data, and since these improvements are independent, their combination boost the performance even bigger.

Let's look into the different text data sources that we used to generate features and study their relative effectiveness. Table 5 summarizes the results of test runs with different combinations of data sources used to generate candidate answer ranking features.

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

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.

Since we used each data source to generate multiple different features for candidate ranking, it is interesting to see which particular features are found more useful than other in the ranking machine learning model. Figure 5 plots different features ranked by their Gini index based feature importances in the final answer candidate ranking random forest

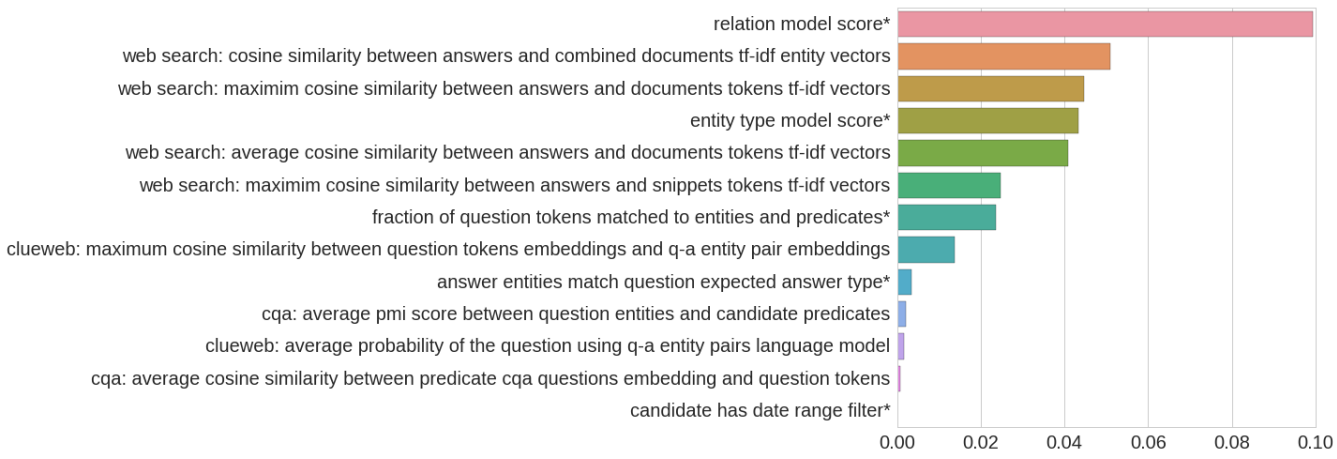

Figure 5: Importances of different text-based features for KBQA (features with * are not text-based and are provided for comparison)

Table 6: Evaluation study for our system with different text-based data sources used to generate features

System	avg Re	avg Pr	avg F1
AQQU	0.604	0.498	0.494
Text2KB -TF	0.6429	0.5030	0.5220
Text2KB +W+CQA+CW+T-E	0.6351	0.4933	0.5104
Text2KB +W+CQA+CW-E	0.6414	0.4981	0.5160
Text2KB +T-W-CQA-CL-E	0.6131	0.4747	0.4918

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

We also manually worked though wins and losses of Text2KB compared to the baseline system. Below we provide some examples, that demonstrate advantages and weaknesses of our approach.

The first set of improvements come from the date range filter template, *e.g.* for the question *"who is the current leader of france 2010?"* our system returns a single correct result *"Nicolas Sarkozy"* instead of the list of all French presidents. The type model score feature helped in some cases, where there is a clear indication of the type of entity, expected as the answer, *e.g. "which state did anne hutchinson found?"* - *"Rhode Island"*.

There are a number of cases when question entity identification using web search results helped to find the right KB entity, which wasn't detected from the question text only, *e.g.* the question *"what did romo do?"* mentions only the last name of the Dallas Cowboys quarterback, whereas web search results mentions the full name multiple times. However, there are cases when additional entities actually made the system to return an incorrect answer, *e.g.* for the question *"what was lucille ball?"* Text2KB added the entity *"I Love Lucy"*, and the candidate answer seeded from this entity got selected as the answer. We should note, that we used a simple entity linking algorithms and strategy to extend question entities, namely we extend the question entity list if term from web search results entity name have high similarity to a term in the question. A better strategies would probably fix the above mentioned problem.

Finally, below are some examples, improved by the proposed web search results features. For the question *"what did bruce jenner win gold medal for?"* the baseline system answered *"1976 Summer Olympics"*, but web search results mention decathlon many times and thus Text2KB was able to rerank the candidates and return the entity *"Athletics at the 1976 Summer Olympics - Men's Decathlon"*[14]. Another interesting example is the question *"what ship did darwin sail around the world?"*, which actually is hard to answer correctly because the ship entity is connected to the Charles Darwin entity through the `user.lindenb.default_domain.scientist.kn` predicate along with some other entities like *"Natural selection"*. There is no predicate, and therefore no such RDF triple, that tells directly what kind of ship did Charles Darwin use. We will see later, that in WebQuestions dataset there is a relatively big number of questions don't have a good match among the predicates. 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, which gave Text2KB enough signal to answer with the ship (along with 2 other unrelated entities also related to *"Charles Darwin"* through the same predicate).

We also found some cases, when our text-based features hurt the performance. For example, the baseline system answered the question *"when did tony romo got drafted?"* correctly, but since almost every mention of *Tony Romo*

[14]Unfortunately, the entity selected as the answer during labeling is *"Decathlon Challenge"*, which is a book Bruce Jenner wrote

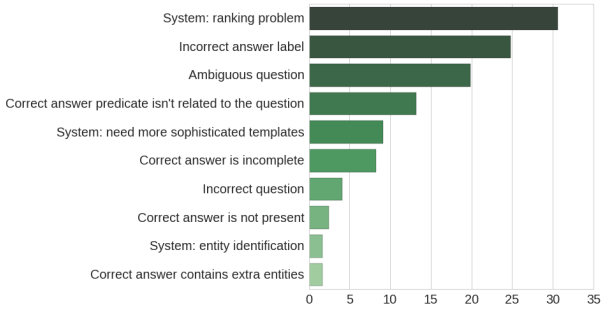

Figure 6: Distribution of problems with examples, where Text2KB returns an answer with F1≠1.0

follows with *Dallas Cowboys*, Text2KB reranked the team name higher and returned it as the answer.

WebQuestions dataset isn't free of noise and quite a few answers are actually incorrect for various reasons. When labeling the question "what team does jordan own?" mechanical turk workers had to select the answer from the page, corresponding to the country and not *Michael Jordan* the basketball player.

5.1 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. We looked through a couple of hundreds of these examples manually and grouped the problems into several typical cases. Figure 6 summarizes our findings.

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 in almost a quarter of missed examples. For example: for the question *when tupac was shot?"* the label says *"Tupac 1994 assault"* instead of *"Las Vegas"* as the ground truth. There is another group of examples, that are related to the problem in the ground truth, *i.e.* incomplete or overcomplete answers. A typical examples of the former are questions which ask to list books, songs or landmarks. The correct answer typically contains ~ 10 entities, but the actual number of correct entities is often larger. It seems to us, that this is an artifact of the labeling process, where the ground truth answer was selected from the Freebase entity profile page, where for long lists of entities only a sample of 10 entities are shown and the other are hidden behind the "NNN values total" link.

There is a certain number of ambiguous questions, *i.e.* vague questions that can be answered with multiple different predicates and there is no obvious reason to prefer one to another. 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 are answered by profession, position a person occupied or sometimes some other achievements. A related problem is when an entity (or Freebase in general) doesn't contain an RDF triple with the predicate that is directly related to 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"* is retrieved through the `olympics.olympic_participating_country.athletes` predicate. In some cases there are entities that are very similar in meaning, but represented with different ids and even names in Freebase. For example, the answer to the question *"what is william taft famous for?"* is *"President of the United States"*, which is a government position, but there is also a triple `[William Howard Taft, common.topic.notable_for, US President]`, where the last entity represents a type of people who help the position, and is considered incorrect.

We also noticed, that in a small number of examples, the case of the correct answer entity and the answer returned by the system is different. For example, for the question *"what fma stands for?"* the correct answer specified in the dataset is *"FullMetal Alchemist"*, while the actual name of the entity is *"Fullmetal Alchemist"*. The official evaluation script don't normalize the case and therefore considers this example incorrect. If we lowercase all entity names before comparison, the average F1 score of Text2KB becomes 0.5248.

GIVE AN ANALYSIS OF PROBLEMS WITH RANKING!!!

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.

In 2011 a series of QALD (Question Answering over Linked Data) evaluation campaigns has started, report on the last one, QALD-5, can be found in [20]. These benchmarks target dbPedia as a knowledge base and provide a rather small (50-350) training set of questions, annotated with correct SPARQL queries. In QALD-3 a multilingual task has been introduced, and since QALD-4 it includes the hybrid task, that 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, however, there are a couple of key differences: unlike this work the hybrid task questions are manually created so that they could *only* be answered by combining RDF and text data. The second difference lies in the nature of text data used for question answering. In this work we use a broad spectrum of various text resources, such as web search results, CQA archives and semantically enriched text documents, while the focus of the task is on relatively short abstracts. Lastly, because of the expensive labeling process QALD datasets are small, for example, QALD-5 training set for multilingual question answering included 300 examples and hybrid - 40 examples (the evaluation was done on 50 questions for multilingual task and just 10 for hybrid). The scale of the datasets makes it hard to use machine learning, which lies in the heart of this work. For these reasons, we didn't include QALD in our evaluation experiments. However, we acknowledge the importance of QALD and intend to look into it and especially into the hybrid track in the future.

WebQuestions benchmark dataset was developed in [5] and since triggered a number of works that explored both semantic parsing and information extraction approaches for KBQA [23]. Developed system differ in ways question entity is detected, candidate answers are generated, features used and the answer is selected. The most interesting aspect for this work is the use of external resources, that help translate natural language text into KB properties and en-

tities. Such a lexicon can be learned from a labeled training set [5], ClueWeb collection aligned to Freebase [24], question paraphrases clusters from WikiAnswers [6], Freebase triples rephrased as questions [9], and can be based on the embeddings of questions and knowledge base entities and predicates [9, 25]. However, most of the models are still biased towards the types of questions present in the training set and would benefit from more training data. In this work I propose to extend the training set with question-answer pairs available on CQA websites, which were shown to be useful for relation extraction [17]. In addition, I propose to use unlabeled text resources for candidate query ranking, which can help to generalize to unseen types of questions and questions about predicates never mentioned in the training set.

In general, combination of 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], *e.g.* IBM! Watson combined text and structured data resources in their Jeopardy! winning system [2]. The key difference of our approach is that such hybrid systems typically have separate pipelines to generate candidate answers from different data sources and only rank them together to select the best one. In this work we attempt to integrate text resources inside the KBQA answer generation and scoring.

Another interesting attempt to combine the data sources has been made by [19], who showed that entity types and descriptions can be effectively used by text-based question answering systems. In this work, we focus on the inverse problem of improving knowledge base question answering by incorporating natural language text resources. ..... MENTION JEFF DALTON'S WORK ON ENTITY ENRICHMENT [**?**].

Open IE extractions [13] represent an interesting mixture between text and structured data and can be considered as an intermediate step between raw text and structured KB. Such knowledge repositories can be queried using structured query languages such as SPARQL and at the same time allows text matching against entities as predicates. One can easily transform an existing KB to such a form by replacing predicates and entities with their names and [14] demonstrated that even though such an approach looses to existing baselines on WebQuestions dataset, where questions are known to be answerable from KB, achieves higher scores on TREC and WikiAnswers benchmarks.

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. ACKNOWLEDGMENTS

Thanks to anonymous reviews for accepting this paper.

9. 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] Q. Cai and A. Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL (1)*, pages 423–433. Citeseer, 2013.

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

[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] V. Murdock and W. B. Croft. A translation model for sentence retrieval. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 684–691, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[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.