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 and ranking. Starting with the best publicly available system, Text2KB utilizes web search results, community question answering and general text document collection data, to detect question topic entities, map question phrases to KB predicates and 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 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 [19], a setting on which we focus in this work.

Two relatively separate approaches for Question Answering (QA) have emerged: text-centric, or Text-QA and knowledge base-centric, or KBQA. In the more traditional, Text-QA approach, systems used text document collections to retrieve passages relevant to a question and extract candidate answers [14]. Unfortunately, an unstructured text passage does not provide explicit information about the candidate entities, which has to be inferred from the context. The KBQA approach, which evolved from the database community, relies on large scale knowledge bases, such as dbPedia [1], Freebase [9] and WikiData [24], 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 KBQA 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 KBQA systems must address three challenges, namely question entity identification to anchor the query process; candidate answer generation; and 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 [28], because it seeds the answer search process. Question text is often quite short, may contain typos and other problems, that complicate entity linking. Existing approaches are usually based on dictionaries that contain entity names, aliases and some other phrases, used to refer to the entities [21]. These dictionaries are 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`. If a dictionary doesn't contain a mapping *"tut"* $\rightarrow$ `Tutankhamun`, as happens for one of the state of the art systems, a system will not be able to answer the question correctly. Such less popular name variations are often used along with full names inside text documents, for example, to avoid repetitions. Therefore, we propose to look into web search results to find variations of question entity names, which can be easier to link to a KB (Figure 2). This idea has been shown

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

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

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

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

effective in entity linking for web search queries[1] [12].

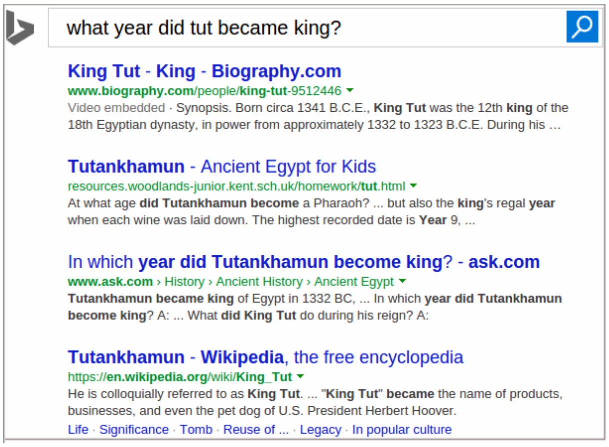

Figure 2: Search results for the question *"what year did tut became king?"*, which mention both the full name of the king and the correct answer to the question

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

To summarize, our contributions are three-fold:

- A novel "hybrid" knowledge base question answering system, which uses both structured data from a knowledge base and unstructured text resources. Section 3 describes the architecture of our system, and Section 4 shows that this fusion improves the performance of a state of the art KBQA system.
- Novel data sources and techniques for KBQA: enhancing question entity identification 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 KBQA worlds, demonstrating an important step forward towards combining unstructured and structured data for question answering.

2. AN OVERVIEW OF KNOWLEDGE BASE QUESTION ANSWERING

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

Recently, KBQA systems have converged to two major approaches: *semantic parsing*, and *information extraction* (IE) [29]. The former focuses on question understanding,

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

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

Interestingly, one of the reasons for recent resurgence of interest in KBQA can be credited to creation of the WebQuestions dataset [6], which is large enough to allow both comprehensive evaluation, and training machine learning methods. Thus, the performance of KBQA systems has quickly improved, with the current state of the art systems using the IE approach, with sophisticated ranking and matching post-processing [31]. In this work, we chose to extend an existing informtion extraction KBQA system – Aqqu [3] – which achieves one of the highest scores among publicly available systems. However, as we will show, our approach is general and can be incorporated into other IE-based systems as well.

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

2.1 The Aqqu KBQA system

First, a KBQA system needs to identify question entities, which are used as sources for the answer search process. For concreteness, consider a question from the WebQuestions dataset *"who is the woman that john edwards had an affair with?"*. In this 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 racing driver (`/m/06zs089`) and other people with the same name. Aqqu considers all spans of question words under certain conditions on part of speech tags and uses a dictionary of names, aliases and anchor texts [21] to map phrases to potential entities. Most recent systems, including Aqqu, don't disambiguate entities at this stage and keep a set of candidates along with some information about their popularities (number of triples in the KB, or number of mentions in the collection) and mention scores $p(entity|mention\ text)$.

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

Each query candidate is represented with a set of features, that includes the scores for linked question entities, various scores for matching between question term n-grams and query predicates, the size of the results list, *etc.* [3]. The final stage of the question answering process is filtering and ranking, using a random forest model, built on the training part of the WebQuestions dataset.

2.2 Basic system extensions

Before introducing our text-based improvements, described in Section 3, we introduced some basic improvements to the original Aqqu system. First, we noticed that since Aqqu does not use information about answer entity Freebase types, in many cases it returns an answer that is incompatible with the question: *e.g.,* state instead of county *etc.* Therefore, we a trained a model to return a score, measuring a compatibility between the question and answer entities, based on the entity notable types and question uni- and bigrams as features, similar to Aqqu's relations score model. A second extension introduced a new date range query template, which 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 to figure out in which range does the specified date falls.

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

3. TEXT2KB: INCORPORATING TEXT DATA INTO KBQA

We now introduce our system, called Text2KB, that expands upon the basic KBQA model by incorporating external textual sources throughout the QA process. The general architecture and an example use case of Text2KB is presented on Figure 3. The left part of the figure roughly corresponds to the architecture of existing information extraction approaches to KBQA. The right part introduces additional external text data sources, specifically we investigate the use of web search results, community question

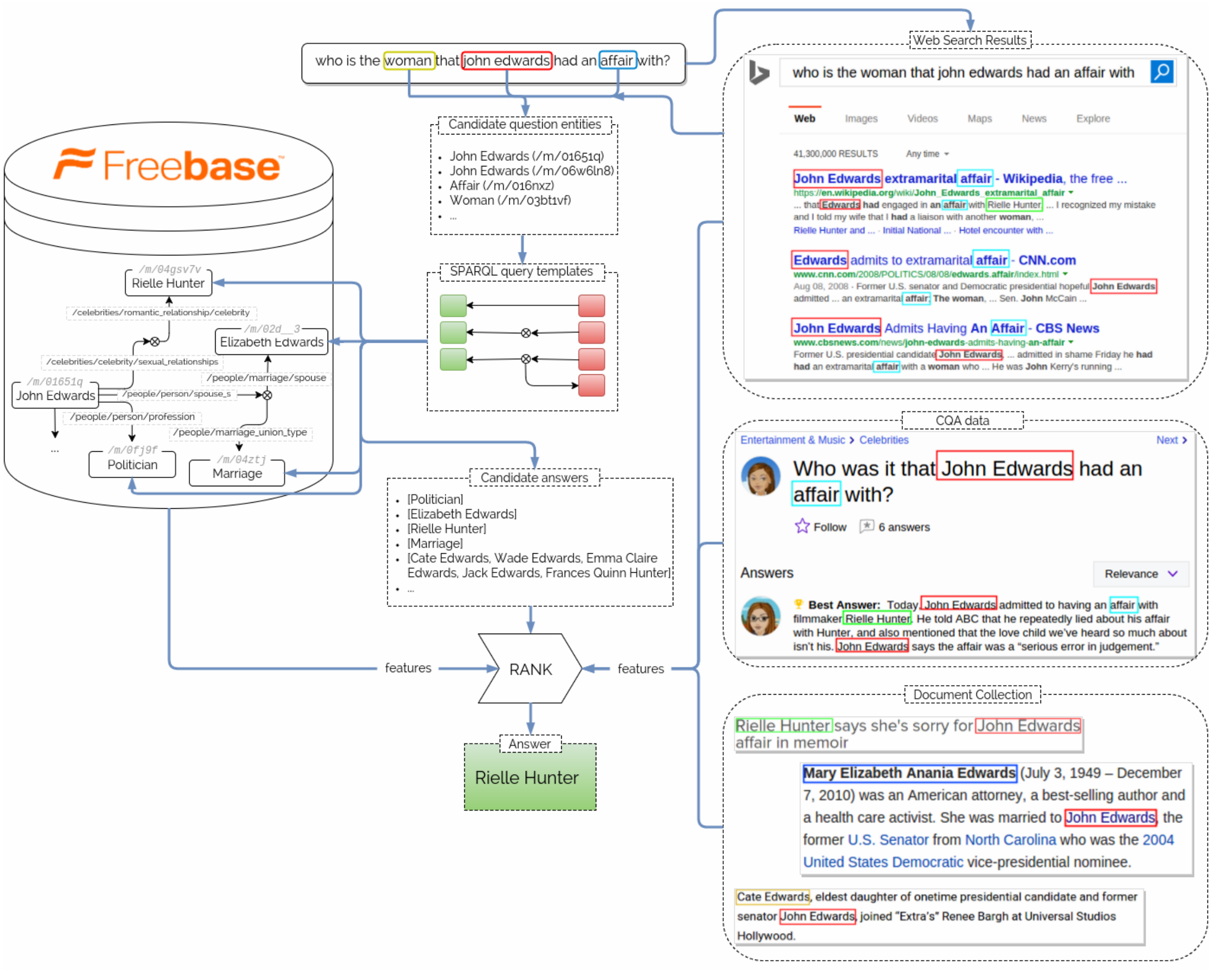

Figure 3: The architecture of our Text2KB Question Answering system

answering (CQA) data, and a collection of documents with detected KB entity mentions. We demonstrate how these data sources can help with the main challenges in KBQA, *i.e.,* question topical entity identification, predicate scoring and answer candidates ranking.

3.1 Web search results for KBQA

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

To obtain related web search results, Text2KB issues the question as a query to a commercial web search engine[2], extracts top 10 result snippets and the corresponding documents. Next, it detects KB entity mentions in both snippets and documents using the same method it applies to the question itself.

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

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

Answer candidate features. The information stored in KBs can also be present in other formats, *e.g.,* text statements. For example, on Figure 2 multiple snippets mention the date when Tutankhamun became the king. Text-QA systems use such passages to extract answer to users' questions. However, text provides very little context information

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

about the mentioned entities, and systems have to infer the useful details, *e.g.,* entity types, which can be problematic [31]. On the other hand, KBQA systems can utilize all the available KB knowledge about the entities in a candidate answer, and would benefit from additional text-based information to improve ranking. More specifically, we perform the following:

1. Precompute term and entity IDFs. We used Google n-grams corpus to approximate terms IDF by collection frequencies and available ClueWeb Freebase entity annotations[3] 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 CQA data for Matching Questions to Predicates

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

For our experiments we use 4.4M questions from Yahoo! WebScore L6 dataset[4]. Question and answer texts were run through an entity linker, that detected mentions of Freebase entities. Next, we use distant supervision assumption to label each question-answer pair with predicates between entities mentioned in the question and in the answer. This

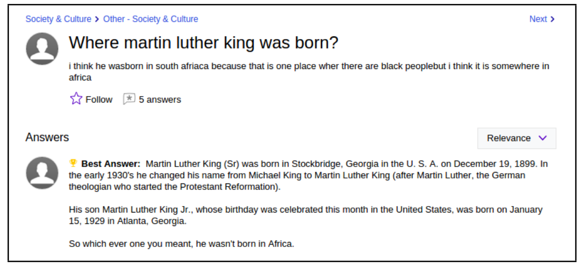

Figure 4: Example of a question and answer pair from Yahoo! Answers CQA website

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

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

labels are used to learn associations between question terms and predicates by computing pointwise mutual information scores (PMI) for each term-predicate pair. Examples of scores for some terms are given in Table 1.

In Text2KB we take candidate answer predicates and look up the PMI scores between them and terms in the question (missing pairs are given a score of 0). From this list of score we compute minimum, average and maximum and add these values to the feature list. Since this kind of data is usually sparse, we also use pretrained word2vec word embeddings[5]. We compute predicate embeddings by taking a weighted average of term vectors from predicate's PMI table. Each term vector is weighted by its PMI value (terms with negative score are skipped). Then, we compute cosine similarities between predicate vector and each of the question term vectors and take their minimum, average, maximum as features. Finally, we average embeddings of question terms and compute its cosine similarity with the predicate vector.

3.3 Estimating Entity Associations

A key step for ranking candidate answers is to estimate whether the question and answer entities are related in a way asked in the question. Existing KBQA approaches usually focus on scoring the mappings between question phrases and KB concepts from a candidate SPARQL query. However, textual data can provide another angle on the problem, as question and answer entities are likely to be mentioned together somewhere in text passages. For example, in the bottom right corner of Figure 3 we can see some passages that mention a pair of people, and the context of these men-

[3]http://lemurproject.org/clueweb09/FACC1/
[4]https://webscope.sandbox.yahoo.com/catalog.php?datatype=l
[5]https://code.google.com/p/word2vec/

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

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

tions explains the nature of the relationships. This data can be viewed as additional edges in a KB, that connect pairs of entities, and have associated language models, estimated from text phrases, that mention these entities. Such edges do not have to coincide with the existing KB edges, and can connect arbitrary pairs of entities, that are mentioned together in text, therefore extending the KB.

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

We use this data to compute candidate ranking features in the following way. Let us have question words Q and an answer candidate, which contains a question entity e_1 and one or more answer entities e_2. For each answer we compute a language model score:

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

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

3.4 Internal text data to enrich entity representation

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

4. EVALUATION

We followed the standard evaluation procedure for the WebQuestions dataset and used the original 70-30% train-test split (3,778 training and 2,032 test instances), and F1-score as the evaluation metric:

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

where $precision(a^*, a) = \frac{|a^* \cap a|}{|a|}$ and $recall(a^*, a) = \frac{|a^* \cap a|}{|a^*|}$, a^* and a are correct and given answers, which can be lists of entities.

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. Modern search engines are very complex and usually include special components to deal with issues, similar to those described in this work, *e.g.,* entity identification. Therefore, to demonstrate that the improvements of Text2KB doesn't simply come from using a commercial search engine, we include a variation of our system that uses Lucene search engine over english Wikipedia.

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

Table 4: Average Recall (R), Precision (Pr), and F1 of Aqqu (baseline), Text2KB (our system), and variations of TextKB with respective components removed. * indicates significant differences at p<0.05.

System	R	Pr	F1
`Aqqu` (baseline)	0.604	0.498	0.494
`Text2KB -E-W-CQA-CL`	0.617	0.481	0.499
`Text2KB -W-CQA-CL`	0.627*	0.492*	0.508*
`Text2KB -E`	0.634*	0.497*	0.514*

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 of 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 it is removed. For example, the baseline system will be denoted as "`Aqqu`". The same system with additional date range filter query templates and notable types score model is denoted as "`Aqqu +DF+T`", which represents the same system as "`Text2KB -E-W-CQA-CL`". Our full system "`Text2KB`" can be also denoted as "`Aqqu +DF+T+E+W+CQA+CL`".

First, let's see what are the improvements introduced by different components of our system (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.

Table 3: Performance of the Text2KB system on WebQuestions dataset compared to the existing approaches. The difference from the baseline Aqqu system is significant with p-value < 0.01

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

[a]The reported score is F1@1. Text-based QA systems usually return a ranked list of answers and are evaluated either using ranking metrics or by taking the top candidate.

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

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

Entity identification using web search results (`Text2KB -W-CQA-CL`) improves both precision and recall. An even bigger improvement is achieved by introducing all additional 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 changes (Table 5).

As we can see, all data sources introduce an improvement to the baseline system, and web search results based features turned out to be the most useful.

Next, we will see which particular features are important in our random forest ranking model. Figure 5 plots a subset of features ranked by their Gini index-based importance scores. The figure supports the observation that web search results features are the most useful, however, other text data sources also contribute to the improvement.

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

5. ANALYSIS AND DISCUSSION

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

Table 6: Average Recall (R), Precision (Pr), and F1 for Text2KB (our system), STAGG and their combinations

System	R	P	F1
Our system: Text2KB	0.6354	0.5059	0.5223
STAGG [31]	0.607	0.528	0.525
Text2KB + STAGG	0.5976	0.5343	0.5320
Text2KB + STAGG (oracle)	0.7144	0.5904	0.6056

We took an existing KBQA systems and demonstrated that by combining evidence from knowledge base and external text resources we can boost the performance. A reasonable question is whether the same approach will be helpful to other systems, *e.g.,* the currently best system – STAGG [31]. STAGG differs from our baseline system Aqqu in the components: entity linking algorithm, a set of query templates and ranking methods. Therefore, our approach is complementary and should be helpful for STAGG as well. To support this claim, we made an experiment to combine answers of STAGG and Text2KB. One of the advantages of the former is its set of filters, that restricts list results to entities of certain type, gender, *etc.* Therefore, we combined answers of STAGG and Text2KB using a simple heuristic: we chose to use the answer returned by STAGG if the number of answer entities is less than in the Text2KB answer, otherwise we use the answer of our approach. Table 6 gives the results of the experiment, and as we can see the combination achieves slightly a better average F1 score. Alternatively, we can look at the oracle combination of the systems, which always selects the answer with higher F1, which demonstrate that systems don't make exactly the same mistakes and therefore can be combined. As we can see such a combination results in a performance of 0.6056, which is much higher than either of the systems.

As we mentioned earlier, answers to 112 of the test questions in WebQuestions dataset involve a predicate that weren't observed in the training set, which may be a problem for approaches that rely on a trained lexicon. We evaluated both systems on these questions, and indeed the performance is

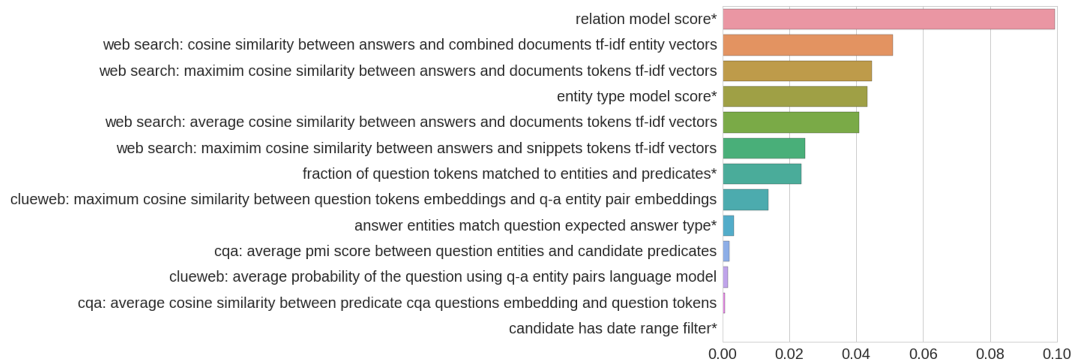

Figure 5: A plot of Gini importances of different features of our answer ranking random forest model (features marked * are not text-based and are provided for comparison)

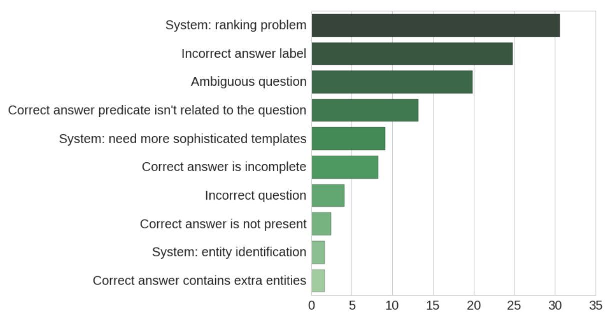

Figure 6: Distribution of problems with questions, where Text2KB returns an answer with F1<1

very low, *i.e.,* the average F1 score of Text2KB is 0.1640 compared to 0.1199 for STAGG[6].

5.1 Error analysis

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

As we can see candidate ranking is still the major problem, and it accounts for 31% of the cases. The second most popular problem is incorrect ground truth labels (almost a quarter of errors). For example: for the question *when tupac was shot?"* the label says `Tupac 1994 assault` instead of `Las Vegas`. Another set of questions have incomplete or overcomplete ground truth answer list. Typical examples are questions asking for a list of movies, books, landmarks, *etc.* 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, which shows only a sample of 10 entities, while the rest is hidden behind the "NNN values total" link. About 20% of the questions are ambiguous, *i.e.,* questions have no strict 1-1 correspondence with any of the predicates and can be answered by multiple ones without any obvious preferences. For example, the question *"what did hayes do?"* can be answered by profession, occupied position or some other achievements. Another problem is when there is no predicate that answers the question. For example, the question *"what do people in france like to do for fun?"* doesn't have a good match among the facts stored in Freebase. The ground truth entity `Cycling` comes from the list Olympic sport competitions country participated[7].

Text2KB components were quite effective in resolving some of the problems. Web search results helped identify the right question topical entity in a number of cases, *e.g., "what did romo do?"* mentions only the last name of the Dallas Cowboys quarterback and the baseline system were unable to map it to the right entity. Web search results provides more than enough evidence that romo refers to `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.,* [12]. An interesting example, when external text resources improved the performance is the question *"what ship did darwin sail around the world?"*. This is actually a hard question, because the ship entity is connected to the `Charles Darwin` entity through the "knownFor" predicate along with some other entities like `Natural selection`. Thus, the predicate itself isn't related to the question, but nevertheless, the name of the ship `HMS Beagle` is mentioned multiple times in the web search results, and entity pair model computed from ClueWeb also has high scores for the terms "ship" and "world".

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

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

[7]`olympics.olympic_participating_country.athletes`

due to large number of mentions in the search results.

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

6. RELATED WORK

One of the most well known benchmarks in knowledge base question answering is QALD (Question Answering over Linked Data) evaluation campaigns, started in 2011 [23]. 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 is included. This task asks participants to use both structured data and free form text available in dbPedia abstracts. The formulation of the hybrid task is the most relevant to our work, but there are 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. 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 for the hybrid task, with 50 and 10 test questions correspondingly. Therefore, due to the scale of datasets and slightly different focus of tasks, we didn't attempt to evaluate our techniques on QALD benchmarks, but intend to explore it in the future.

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

In general, combining different data sources, such as text documents and knowledge bases, for question answering is not a novel idea, and it has been already implemented in hybrid QA systems [5, 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 [22], who explored the use of entity types and descriptions from a KB for text-based question answering, and [13] explored such semantic annotations for ad-hoc document retrieval.

We should also mention Open Information Extraction (Open IE) [15], which is an interesting mixture between text and structured data. Open repositories can be queried using structured query languages, and at the same time allows keyword matching against entities and predicates [16]. In this work, we are borrowing an idea of learning about entity relationship via natural language phrases connecting them. However, since we don't need to extract clean set of relation tuples, we can keep all kinds of phrases, mentioned around entity pairs.

M. Yahya et al [27] proposed an interesting idea of extending SPARQL triple patterns with text keywords, used certain query relaxation techniques to improve the robustness of KBQA systems. Query relaxation is dropping certain triples patterns from SPARQL query and adding the corresponding question words as keywords to other triple patterns. The idea of query relaxations and using text in SPARQL queries was extended in [26], which proposed a framework for querying extended knowledge graph, comprising of a combination of KB and OpenIE triples. These ideas are complimentary to our work, because our use of text data improves the matching between question phrases and KB concepts, whereas query relaxations are applied when a good match wasn't found. Another KB-Text hybrid approach, proposed in [25], utilizes text resources as a post processing step for answer validation and filtering.

7. CONCLUSIONS AND FUTURE WORK

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

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

8. ACKNOWLEDGMENTS

This work was funded by ???????. We gratefully thank the anonymous reviewers for their constructive comments.

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. CIKM '15, New York, NY, USA, 2015.

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

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

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

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

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

[9] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM International Conference on Management of Data*, New York, NY, USA, 2008.

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

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

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

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

[14] H. T. Dang, D. Kelly, and J. J. Lin. Overview of the trec 2007 question answering track. In *TREC*, volume 7, page 63. Citeseer, 2007.

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

[16] A. Fader, L. Zettlemoyer, and O. Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of SIGKDD*, KDD '14, New York, NY, USA, 2014.

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

[18] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL 2009*, pages 1003–1011. Association for Computational Linguistics, 2009.

[19] J. Pound, P. Mika, and H. Zaragoza. Ad-hoc object retrieval in the web of data. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 771–780, New York, NY, USA, 2010. ACM.

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

[21] V. I. Spitkovsky and A. X. Chang. A cross-lingual dictionary for english wikipedia concepts. In *Proceedings of LREC'12*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

[22] H. Sun, H. Ma, W.-t. Yih, C.-T. Tsai, J. Liu, and M.-W. Chang. Open domain question answering via semantic enrichment. WWW '15, Republic and Canton of Geneva, Switzerland, 2015.

[23] C. Unger, C. Forascu, V. Lopez, A.-C. N. Ngomo, E. Cabrio, P. Cimiano, and S. Walter. Question answering over linked data (qald-5). In *CLEF (Working Notes)*, volume 1391 of *CEUR Workshop Proceedings*, 2015.

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

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

[26] M. Yahya, D. Barbosa, K. Berberich, Q. Wang, and G. Weikum. Relationship queries on extended knowledge graphs. In *Proceedings of the ACM International Conference on Web Search and Data Mining*. ACM, 2016.

[27] M. Yahya, K. Berberich, S. Elbassuoni, and G. Weikum. Robust question answering over the web of linked data. In *Proceedings of the ACM international conference on information & knowledge management*. ACM, 2013.

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

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

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

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