Künstliche Intelligenz manuscript No.
(will be inserted by the editor)

A Quality Evaluation of Combined Search on a Knowledge Base and Text

Hannah Bast · Björn Buchhold · Elmar Haussmann

Received: date / Accepted: date

Abstract We provide a quality evaluation of KB+Text search, a deep integration of knowledge base search and standard full-text search. A knowledge base (KB) is a set of subject-predicate-object triples with a common naming scheme. The standard query language is SPARQL, where queries are essentially lists of triples with variables. KB+Text extends this search by a special *occurs-with* predicate, which can be used to express the co-occurrence of words in the text with mentions of entities from the knowledge base. Both pure KB search and standard full-text search are included as special cases.

We evaluate the result quality of KB+Text search on three different query sets. The corpus is the full version of the English Wikipedia (2.4 billion word occurrences) combined with the YAGO knowledge base (26 million triples). We provide a web application to reproduce our evaluation, which is accessible via `http://ad.informatik.uni-freiburg.de/publications`.

Keywords Knowledge Bases · Semantic Search · KB+Text Search · Quality Evaluation

1 Introduction

KB+Text search combines structured search in a knowledge base (KB) with traditional full-text search.

In traditional full-text search, the data consists of text documents. The user types a (typically short) list of keywords and gets a list of documents containing some or all of these keywords, hopefully ranked by some notion of relevance to the query. For example, typing *broccoli leaves edible* in a web search engine will return lots of web pages with evidence that broccoli leaves are indeed edible.

In KB search, the data is a knowledge base, typically given as a (large) set of subject-predicate-object triples. For example, *Broccoli is-a plant* or *Broccoli native-to Europe*. These triples can be thought of to form a graph of entities (the nodes) and relations (the edges), and a language like SPARQL allows to search for subgraphs matching a given pattern. For example, find all plants that are native to Europe.

The motivation behind KB+Text search is that many queries of a more "semantic" nature require the combination of both approaches. For example, consider the query *plants with edible leaves and native to Europe*, which will be our running example in this paper. A satisfactory answer for this query requires the combination of two kinds of information:

(1) A list of plants native to Europe; this is hard for full-text search but a showcase for knowledge base search.

(2) For each plant the information whether its leaves are edible or not; this is easily found with a full-text search for each plant, but quite unlikely to be contained in a knowledge base.

In a previous work [4], we have developed a system with a convenient user interface to construct such

Department of Computer Science
University of Freiburg
79110 Freiburg, Germany
Tel.: +49 761 203-8163
E-mail: {bast,buchhold,haussmann}@cs.uni-freiburg.de

queries incrementally, with suggestions for expanding the query after each keystroke. We named the system Broccoli, after a variant of the example query above, which was our very first test query. Figure 1 shows a screenshot of Broccoli in action for this example query.

1.1 Our Contribution

The main contribution of this paper is a quality evaluation of KB+Text search on three benchmarks, including a detailed error analysis; see Section 4. On the side, we recapitulate the basics of KB+Text search (Section 2) and we provide a brief but fairly broad overview of existing quality evaluations for related kinds of "semantic search" (Section 3).

2 The Components of KB+Text Search

We briefly describe the main components of a system for KB+Text search, as far as required for understanding the remainder of this paper. KB+Text search operates on two kinds of inputs, a knowledge base and a text collection. The knowledge base consists of entities and their relations in the form of triples. The text collection consists of documents containing plain text. This input is pre-processed, indexed, and then queried as follows.

2.1 Entity Recognition

In a first step, mentions of entities from the given knowledge base are recognized in the text. Consider the following sentence:

(S) *The usable parts of rhubarb are the medicinally used roots and the edible stalks, however its leaves are toxic.*

Assuming the provided knowledge base contains the entity *Rhubarb*, the words *rhubarb* and *its* are references to it. When we index the English Wikipedia and use YAGO as a knowledge base, we make use of the fact that first occurrences of entities in Wikipedia documents are linked to their Wikipedia page that identifies a YAGO entity. Whenever a part or the full name of that entity is mentioned again in the same section of the document (for example, *Einstein* referring to *Albert Einstein*), we recognize it as that entity. We resolve references (anaphora) by assigning each occurrence of *he*, *she*, *it*, *her*, *his*, etc. to the last recognized entity of matching gender. For text without Wikipedia annotations, state-of-the art approaches for named entity recognition and disambiguation, such as Wikify! [20], can be used instead.

2.2 Text Segmentation

The special *occurs-with* relation searches for co-occurrences of words and entities as specified by the respective arc in the query; see Figure 1 and Section 2.4 below. We identify segments in the input text to which co-occurrence should be restricted. Identifying the ideal scope of these segments is non-trivial and we experiment with three settings: (1) complete sections, (2) sentences and (3) *contexts*, which are parts of sentences that "belong together" semantically. The contexts for our example sentence (S) from above are:

(C1) *The usable parts of <u>rhubarb</u> are the medicinally used roots*
(C2) *The usable parts of <u>rhubarb</u> are the edible stalks*
(C3) *however <u>rhubarb</u> leaves are toxic*

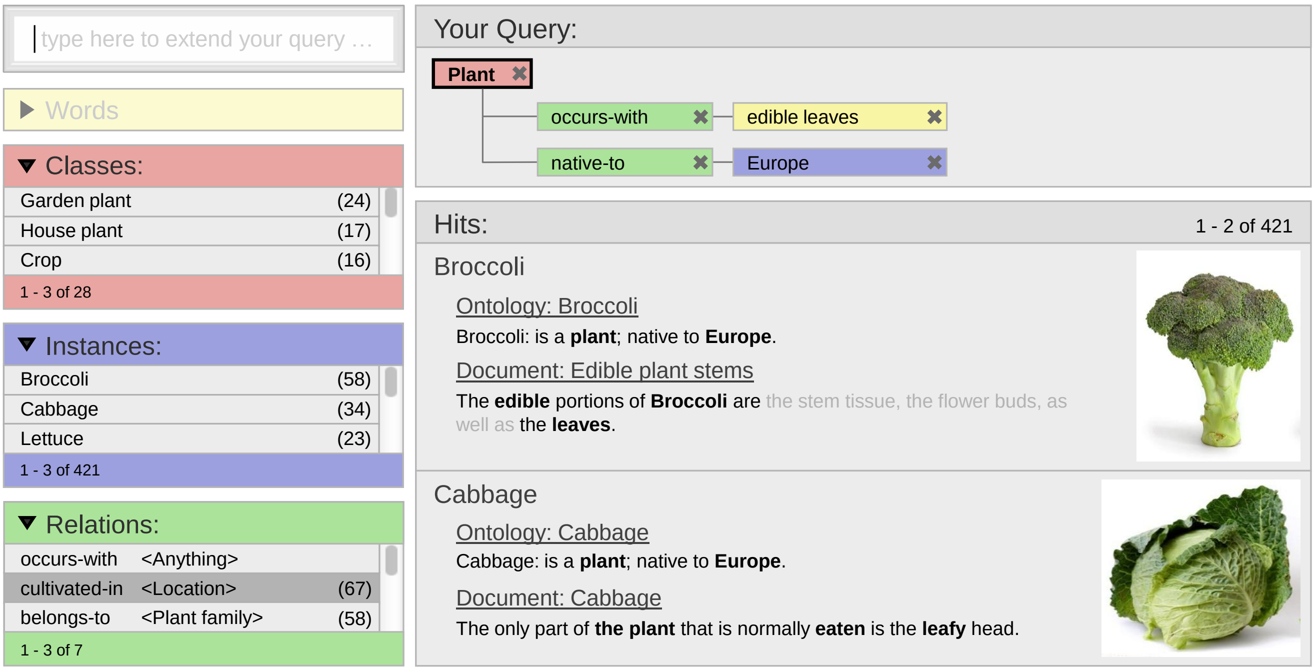
Note that, because entities and references (underlined words) have been identified beforehand, no information is lost. The rationale behind contexts is to make the search more precise and "semantic". For example, we would not want *Rhubarb* to be returned for a query for *plants with edible leaves*, since its leaves are actually toxic. Nevertheless *Rhubarb*, *edible*, and *leaves* co-occur in sentence (S) above. However, they do not co-occur in either of (C1), (C2), (C3). To compute contexts, we follow an approach for Open Information Extraction (OIE) described in [8].

2.3 Indexing

An efficient index for KB+Text search is described in [5]. This index provides exactly the support needed for the system shown in Figure 1: efficient processing of tree-shaped KB+Text queries (without variables for relations), efficient excerpt generation, and efficient search-as-you-type suggestions that enable a fully interactive incremental query construction.

2.4 Query Language

KB+Text extends ordinary KB search by the special *occurs-with* predicate. This predicate can be used to specify co-occurrence of a class (e.g., plant) or instance



Classes:	
Garden plant	(24)
House plant	(17)
Crop	(16)
1 - 3 of 28	

Instances:	
Broccoli	(58)
Cabbage	(34)
Lettuce	(23)
1 - 3 of 421	

Relations:		
occurs-with	<Anything>	
cultivated-in	<Location>	(67)
belongs-to	<Plant family>	(58)
1 - 3 of 7		

Ontology: Broccoli

Broccoli: is a **plant**; native to **Europe**.

Document: Edible plant stems

The **edible** portions of **Broccoli** are the stem tissue, the flower buds, as well as the **leaves**.

Ontology: Cabbage

Cabbage: is a **plant**; native to **Europe**.

Document: Cabbage

The only part of **the plant** that is normally **eaten** is the **leafy** head.

Fig. 1 A screenshot of Broccoli, showing the final result for our example query. The box on the top right visualizes the current KB+Text query as a tree. The large box below shows the matching instances (of the class from the root node, plant in this case). For each instance, evidence is provided for each part of the query. In the panel on the left, instances are entities from the knowledge base, classes are groups of entities with the same object according to the *is-a* predicate, and relations are predicates. The instances are ranked by the number of pieces of evidence (only a selection of which are shown). With the search field on the top left, the query can be extended further. Each of the four boxes below the search field provide context-sensitive suggestions that depend on the current focus in the query. For the example query: suggestions for sub*classes* of plants, suggestions for *instances* of plants that lead to a hit, suggestions for *relations* to further refine the query. Word suggestions are displayed as soon as the user types a prefix of sufficient length. These suggestions together with the details from the hits box allow the user to incrementally construct adequate queries without prior knowledge of the knowledge base or of the full text.

(e.g., Broccoli) with an arbitrary combination of words, instances, and further sub-queries. When processing the query, this co-occurrence is restricted to the segments identified in the pre-processing step described in Section 2.2 above.

A user interface, like the one shown in Figure 1, guides the user in incrementally constructing queries from this language. In particular, a visual tree-like representation of the current query is provided after each keystroke, along with hits for that query and suggestions for further extensions or refinements.

3 Related Work

The literature on semantic search technologies is vast, and “semantic” means many different things to different researchers. A variety of different and hard-to-compare benchmarks have therefore emerged, as well as various stand-alone evaluations of systems that perform KB+Text or variants of it.

We briefly discuss the major benchmarks from the past decade, as well as the relatively few systems that explicitly combine full-text search and knowledge base search. A comprehensive survey of the broad field of semantic search on text and/or knowledge bases is provided in [6].

3.1 TREC Entity Tracks

The goal of the TREC Entity Tracks were queries searching for lists of entities, just like in our KB+Text search. They are particularly interested in lists of entities that are related to a given entity in a specific way. Thus, the task is called ”related entity finding”. A typical query is *airlines that currently use boeing 747 planes*. Along with the query, the central entity (*boeing 747*) as well as the the type of the desired target entities (*airlines*) were given.

For the 2009 Entity Track [3], the underlying data was the ClueWeb09 category B collection. ClueWeb09[1] is a web corpus consisting of 1 billion web pages, of which 500 million are in English. The category B collection is a sub-collection of 50 million of the English pages. Runs with automatic and manual query construction were evaluated. This task turned out to be very hard, and the overall best system achieved an NDCG@R of 31% and a P@10 of only 24% - albeit with automatic query construction. When restricting the results to entities from Wikipedia, the best system achieved an NDCG@R of 22% and a P@10 of 45% [12]. We use the queries from this track as one of our benchmarks in Section 4 (for later tracks no Wikipedia based groundtruth is available).

For the 2010 Entity Track [1], the full English portion of the ClueWeb09 dataset was used (500 million pages). The task remained hard, with the best system achieving an NDCG@R of 37% and an R-Precision (P@10 was not reported that year) of 32% even for manually tuned queries (and 30% for automatic runs).

In 2010, an additional task was added, Entity List Completion (a similar task but with an additional set of example result entities given for each query) with BTC 2009 as the underlying dataset.[2] This is a dataset consisting of 1.14 billion triples crawled from the semantic web. The BTC dataset contains the complete DBpedia [18]. It turned out that the best performing approaches all boost triples from DBpedia to obtain good results. Still, working with the dataset turned out to be difficult, with the best systems achieving an R-Precision of 31% (NDCG@R was not reported).

In the 2011 track [2], another semantic web dataset was used (Sindice 2011 [13]). However, the number of participating teams was very low, and results were disappointing compared to previous years.

3.2 SemSearch Challenges

The task in the SemSearch challenges is also referred to as *ad-hoc object retrieval* [17]. The user inputs free-form keyword queries, e.g. *Apollo astronauts who walked on the moon* or *movies starring Joe Frazier*. Results are ranked lists of entities. The benchmarks were run on BTC 2009 as a dataset.

In the 2010 challenge [17], there were 92 queries, each searching only for a single entity. The best system achieved a P@10 of 49% and a MAP of 19%.

In the 2011 challenge [10], there were 50 queries. The best system achieved a P@10 of 35% and a MAP of 28%. The 2011 queries are one of our benchmarks in Section 4.

3.3 The INEX Series

INEX (Initiative for the Evaluation of XML Retrieval) has featured many search tasks. While the focus is on XML retrieval, two tracks are remarkably similar to the benchmarks discussed before.

The Entity Ranking Track (from 2007 to 2009) and the Linked-Data Track (2012 and 2013) work on the text from Wikipedia and use intra-Wikipedia links to establish a connection between entities and an ontology or an entire knowledge base (since 2012, entities are linked to their representation in DBpedia). Queries are very similar to those of the TREC Entity Track from above: given a keyword query (describing a topic) and a category, find entities from that category relevant for that topic. However, few participants actually made use of linked data in their approaches and the results were inconclusive.

3.4 Question Answering

Question answering (QA) systems provide a functionality similar to KB+Text. The crucial difference is that questions can be asked in natural language (NL), which makes the answering part much harder. Indeed, the hardest part for most queries in the QA benchmarks is to “translate” the given NL query into a query that can be fed to the underlying search engine.

In the TREC QA tracks, which ran from 1999 to 2007, the underlying data were corpora of text documents. An overview of this long series of tracks is given in [15]. The corpora were mainly newswire documents, later also blog documents. The series started with relatively simple factoid questions, e.g. *Name a film in which Jude Law acted*, and ended with very complex queries based on sequences of related queries, including, e.g., temporal dependencies. For list questions, such as *Who are 6 actors who have played Tevye in 'Fiddler*

[1] `http://lemurproject.org/clueweb09/`

[2] BTC = billion triple challenge, `https://km.aifb.kit.edu/projects/btc-2009/`

on the Roof'?, which are similar to the kind we consider in this paper, the best system in 2007 achieved an F-measure of 48%.

In the QALD (Question Answering over Linked Data) series of benchmarks [19], the underlying data is again a large set of fact triples. The task is to generate the correct SPARQL query from a given NL question of varying difficulty, e.g. *Give me all female Russian astronauts* [14]. This is very different from the other benchmarks described above, where a perfect query (SPARQL or keyword) typically does not exist.

The various tracks used different sets of facts triples from DBpedia and MusicBrainz (facts about music). In the last two runs, QALD-4 [24] and QALD-5 [25], the best system achieved an an F-measure of 72% and 63%, respectively.

3.5 Systems for KB+Text and Similar Paradigms

Systems for a combined search in text documents and knowledge bases were previously proposed in [7] (ESTER), [9] (Hybrid Search), [22] (Mímir), [26] (Semplore), and [16] (Concept Search). None of these systems consider semantic context as described in Section 2.2. For all these systems, only a very limited quality evaluation has been provided.

Hybrid Search is evaluated on a very specialized corpus (18K corporate reports on jet engines by Rolls Royce). For Concept Search, a similarly small dataset of 29K documents constructed from the DMoz web directory.

For ESTER, only two simple classes of queries are evaluated: *people associated with <university>* and *list of counties in <US state>*. A precision of 37% and 67%, respectively, is reported for each class.

Semplore is evaluated on a combination of DBpedia (facts from Wikipedia) and LUBM (an ontology for the university domain). A P@10 of more than 80% is reported for 20 manually constructed queries. For many of those, the text part is simply keyword search in entity names, e.g., all awards containing the keywords *nobel prize*. Those queries then trivially have perfect precision and recall. We have only a single such query in our whole quality evaluation, all other queries combine knowledge base and full-text search in a non-trivial manner.

Mímir is only evaluated with respect to query response times and in a user study where users were asked to perform four search tasks. For these tasks, success and user satisfaction with the system were tracked.

4 Evaluation

4.1 Input Data

The text part of our data is all documents in the English Wikipedia, obtained via `download.wikimedia.org` in January 2013.[3] Some dimensions of this collection: 40 GB XML dump, 2.4 billion word occurrences (1.6 billion without stop-words), 285 million recognized entity occurrences and 200 million sentences which we decompose into 418 million contexts.

As knowledge base we used YAGO from October 2009.[4] We manually fixed 92 obvious mistakes in the KB (for example, the *nobel prize* was a *laureate* and hence a *person*), and added the relation *Plant native-in Location* for demonstration purposes. Altogether our variant of YAGO contains 2.6 million entities, 19,124 classes, 60 relations, and 26.6 million facts.

We build a joint index over this full text and this KB, as described in [5]. As described there, the resulting index file has a size of 14 GB with query times typically well below 100 ms [5, Table 1].

4.2 Query Benchmarks

We evaluated the quality of our KB+Text search on the dataset just described on three query benchmarks. Each benchmark consists of a set of queries, and for each query the set of relevant entities for that query from the knowledge base (YAGO in our case). Two of these query benchmarks are from past entity search competitions, described in Section 3: the Yahoo SemSearch 2011 List Search Track [23] and the TREC 2009 Entity Track [3]. The third query benchmark is based on a random selection of ten Wikipedia featured *List of ...* pages, similarly as in [7].

To allow reproducibility, we provide queries and relevance judgments as well as the possibility to evaluate (and modify) the queries against a live running system for the SemSearch List Track and the Wikipedia lists under the link provided in the abstract. The TREC Entity Track queries were used for an in-depth quality evaluation that does not allow for an easy reproduction.

[3] The choice of this outdated version has no significant impact on the insights from our evaluation: the corresponding Wikipedia data from 2017 is (only) about 50% larger but otherwise has the same characteristics and would not lead to principally different results.

[4] There is a more recent version, called YAGO2, but most of the additions from YAGO to YAGO2 (spatial and temporal information) are not interesting for our search.

Therefore we do not provide them in our reproducibility web application.

The SemSearch 2011 List Search Track consists of 50 queries asking for lists of entities in natural language, e.g. *Apollo astronauts who walked on the Moon*. The publicly available results were created by pooling the results of participating systems and are partly incomplete. Furthermore, the task used a subset of the BTC dataset (see Section 3), and some of the results referenced the same entity several times, e.g., once in DBpedia and once in OpenCyc. Therefore, we manually created a new ground truth consisting only of Wikipedia entities (compatible with our dataset). This was possible because most topics were inspired by Wikipedia lists and can be answered completely by manual investigation. Three of the topics did not contain any result entities in Wikipedia, and we ignored one additional topic because it was too controversial to answer with certainty (*books of the Jewish canon*). This leaves us with 46 topics and a total of 384 corresponding entities in our ground truth. The original relevance judgments only had 42 topics with primary results and 454 corresponding entities, including many duplicates.

The TREC 2009 Entity Track worked with the ClueWeb09 collection and consisted of 20 topics also asking for lists of entities in natural language, e.g. *Airlines that currently use Boeing 747 planes*, but in addition provided the source entity (*Boeing 747*) and the type of the target entity (*organization*). We removed all relevance judgments for pages that were not contained in the English Wikipedia; this approach was taken before in [12] as well. This leaves us with 15 topics and a total of 140 corresponding relevance judgments.

As a third benchmark we took a random selection of ten of Wikipedia's over 2,400 `en.wikipedia.org/wiki/List_of_...` pages[5]. For example, *List of participating nations at the Winter Olympic Games*. These lists are manually created by humans, but actually they are answers to semantic queries. The lists also tend to be fairly complete, since they undergo a review process in the Wikipedia community. This makes them perfectly suited for a quality evaluation of our system. For the ground truth, we automatically extracted the list of entities from the Wikipedia list pages. This leaves us with 10 topics and a total of 2,367 corresponding entities in our ground truth.

For all of these tasks we manually generated KB+Text queries corresponding to the intended semantics of the original queries. We relied on the interactive query suggestions of the user interface, but did not fine-tune queries towards the results. We want to stress that our goal is not a direct comparison to systems that participated in the tasks above. For that, input, collection and relevance judgments would have to be perfectly identical. Instead, we want to evaluate whether KB+Text can provide high quality results for these tasks.

4.3 Quality Measures

Table 1 displays set-related and ranking-related measures. Our set-related measures include the numbers of false-positives (#FP) and false-negatives (#FN). We calculate the precision (Prec.) as the percentage of retrieved relevant entities among all retrieved entities and the recall as the percentage of retrieved relevant entities among all relevant entities. We calculate the F-measure (F1) as the harmonic mean of precision and recall.

For our ranking-related measures, we ordered entities by the number of matching segments. Let P@k be the percentage of relevant documents among the top-k entities returned for a query. R-precision is then defined as P@R, where R is the total number of relevant entities for the query. The average precision (AP) is the average over all P@i, where i are the positions of all relevant entities in the result list. For relevant entities that were not returned, a precision with value 0 is used for inclusion in the average. The mean average precision (MAP) is then simply the average AP over all queries. We calculate the discounted cumulative gain (DCG) as:

$$DCG = \sum_{i=1}^{\#rel} \frac{rel(i)}{\log_2(1+i)}$$

where $rel(i)$ is the relevance of the entity at position i in the result list. Usually, the measure supports different levels of relevance, but we only distinguish 1 and 0 in our benchmarks. The nDCG is the DCG normalized by the score for a perfect DCG. Thus, we divide the actual DCG by the maximum possible DCG for which we can simply take all $rel(i) = 1$.

4.4 Quality Results

Table 1 evaluates our quality measures for all three benchmarks. As described in Section 2, the key com-

[5] `http://en.wikipedia.org/wiki/Wikipedia:Featured_lists`

Table 1 Sum of false-positives and false-negatives and averages for the other measures over all SemSearch, Wikipedia list and TREC queries for the evaluated system when running on sections, sentences or contexts. The averages for F1, R-Prec, MAP, nDCG are macro-averages over all queries (that is, for example, the F1 in the first row is the average F1 of all SemSearch queries when running on sections). The $*$ and † denote a p-value of < 0.02 and < 0.003, respectively, for the two-tailed t-test compared to the figures for sentences.

		#FP	#FN	Prec.	Recall	F1	R-Prec	MAP	nDCG
SemSearch	sections	44,117	92	0.06	0.78	0.09	0.32	0.42	0.44
	sentences	1,361	119	0.29	0.75	0.35	0.32	0.50	0.49
	contexts	**676**	**139**	**0.39**	**0.67**	**0.43†**	**0.52**	**0.45**	**0.48**
WP lists	sections	28,812	354	0.13	0.84	0.21	0.38	0.33	0.41
	sentences	1,758	266	0.49	0.79	0.58	0.65	0.59	0.68
	contexts	**931**	**392**	**0.61**	**0.73**	**0.64∗**	**0.70**	**0.57**	**0.69**
TREC	sections	6,890	19	0.05	0.82	0.08	0.29	0.29	0.33
	sentences	392	38	0.39	0.65	0.37	0.62	0.46	0.52
	contexts	**297**	**36**	**0.45**	**0.67**	**0.46∗**	**0.62**	**0.46**	**0.55**

ponent of our KB+Text search is the *occurs-with* relation, which searches for co-occurrences of the specified words / entities. We compare three segmentations for determining co-occurrence, as described in Section 2.2: *sections*, *sentences*, and semantic *contexts*.

Compared to sentences, semantic contexts decrease the (large) number of false-positives significantly for all three benchmarks.[6] Using co-occurence on the section level we can observe a decrease in the number of false-negatives (a lot of them due to random co-occurrence of query words in a section). However, this does not outweigh the drastic increase of the number of false-positives. Overall, semantic contexts yield the best precision on all three benchmarks, and also the best F-measure. This confirms the positive impact on the user experience that we have observed.

4.5 Error Analysis

KB+Text search, as described in Section 2 is a complex task, with many potential sources for errors. For the TREC benchmark, using contexts as segments, we manually investigated the reasons for the false-positives and false-negatives. We defined the following error categories.

For false-positives:

(FP1) a true hit was *missing* from the ground truth;
(FP2) the context has the *wrong meaning*;[7]
(FP3) a mistake in the *knowledge base*;
(FP4) a mistake in the *entity recognition*;
(FP5) a mistake by the *parser*;[8]
(FP6) a mistake in *computing contexts*.

For false-negatives:

(FN1) there seems to be *no evidence* for this entity in Wikipedia based on the query we used (the fact might be present but expressed using different words);
(FN2) the query elements are *spread* over two or more sentences;
(FN3) a mistake in the *knowledge base*;
(FN4) a mistake in the *entity recognition*;
(FN5) a mistake by the *parser* (analogous to FP5);
(FN6) a mistake in *computing contexts*.

Table 2 Breakdown of all errors by category.

#FP	FP1	FP2	FP3	FP4	FP5	FP6
297	55%	11%	5%	12%	16%	1%
#FN	**FN1**	**FN2**	**FN3**	**FN4**	**FN5**	**FN6**
36	22%	6%	26%	21%	16%	8%

Table 2 provides the percentage of errors in each of these categories. The high number in FP1 is great

[6] For the TREC benchmark even the number of false-negatives decreases. This is because when segmenting into contexts the document parser pre-processes Wikipedia lists by appending each list item to the preceding sentence. These are the only types of contexts that cross sentence boundaries and a rare exception. For the Wikipedia list benchmark we verified that this technique does not include results from the lists from which we created the ground truth.

[7] This means that the words occur in the context, but with a meaning different from what was intended by the query.

[8] The sentence parses are required to compute contexts.

Table 3 Quality measures for the TREC benchmark for the *orig*inal ground truth, with *miss*ing relevant entities, and with errors from categories FP and FN 3,4,5 *corr*ected.

	Prec.	Recall	F1	P@10	R-Prec	MAP	nDCG
TREC Entity Track, best	n/a	n/a	n/a	0.45	0.55	n/a	0.22
KB+Text, orig	0.45	0.67	0.46	0.58	0.62	0.46	0.55
KB+Text, orig + miss	0.67	0.73	0.65	0.79	0.77	0.62	0.70
KB+Text, orig + miss + corr	0.88	0.89	0.86	0.94	0.92	0.85	0.87

news for us: many entities are missing from the ground truth but were found by the system. Errors in FN1 occur when full-text search with our queries on whole Wikipedia documents does not yield hits, independent from semantic contexts. Tuning queries or adding support for synonyms can decrease this number. FP2 and FN2 comprise the most severe errors. They contain false-positives that still match all query parts in the same context but have a different meaning and false-negatives that are lost because contexts are confined to sentence boundaries. Fortunately, both numbers are quite small.

The errors in categories FP and FN 3-5 depend on implementation details and third-party components. The high number in FN3 is due to errors in the used knowledge base, YAGO. A closer inspection revealed that, although the triples in YAGO are reasonably accurate, it is vastly incomplete in many areas. For example, the *acted-in* relation contains only one actor for most movies. This could be mitigated by switching to a more comprehensive knowledge base like Freebase [11]; indeed, our latest demo of Broccoli is using Freebase instead of YAGO [4]. To mitigate the errors caused by entity recognition and anaphora resolution (FP4+FN4), a more sophisticated state-of-the-art approach is easily integrated. Parse errors are harder. The current approach for determining contexts heavily relies on the output of a state-of-the art constituent parser. Assuming a perfect parse for every single sentence, especially those with flawed grammar, is not realistic. Still, those errors do not expose limits of KB+Text search with semantic contexts. The low number of errors due to the context computation (FP6+FN6) demonstrates that the current approach (Section 2.2) is already pretty good. Fine-tuning the way we decompose sentences might decrease this number even further.

Table 3 provides an updated evaluation, with all the errors induced by "third-party" components (namely FP and FN 3,4,5) corrected. The last row shows the high potential of KB+Text search and motivates further work correcting the respective errors. As argued in the discussion after Table 2, many corrections are easily applied, while some of them remain hard to correct perfectly.

The first line of Table 3 shows the best results from the TREC 2009 Entity Track (TET09), when restricted to entities from the English Wikipedia; see [12, Table 10]. There are a few things to note in this comparison. First, TET09 used the ClueWeb09 collection, category B. However, that collection contains the English Wikipedia, and participants were free to restrict their search to that part only. Indeed, the best systems strongly boosted results from Wikipedia. Second, results for TET09 were not sets but ranked lists of entities, hence absolute precision and recall figures are not available. Our results are for the simplistic ranking explained above. Third, we created our queries manually, as described at the end of Section 4.2 above. However, TET09 also permitted manually constructed queries, but those results were not among the best. Fourth, the ground truth was approximated via *pooling* results from the then participating systems [3]. This is a disadvantage for systems that are evaluated later on the same ground truth [21]. Still, our quality results are better even on the original ground truth, and much better with missing entities (FP1) added.

5 Conclusions and Future Work

We have evaluated the quality of KB+Text search on three benchmarks, with very promising results. A detailed error analysis has pointed out the current weak spots: missing entities in the knowledge base, missing evidence in the full text, errors in the entity recognition, errors in the full parses of the sentences. Promising directions for future research are therefore: switch to a richer knowledge base (e.g., Freebase), switch to a larger corpus than Wikipedia (e.g., ClueWeb), develop a more sophisticated entity recognition, try to determine semantic context without full parses.

References

1. Balog, K., Serdyukov, P., de Vries, A.P.: Overview of the TREC 2010 Entity Track. In: TREC (2010)
2. Balog, K., Serdyukov, P., de Vries, A.P.: Overview of the TREC 2011 Entity Track. In: TREC (2011)
3. Balog, K., de Vries, A.P., Serdyukov, P., Thomas, P., Westerveld, T.: Overview of the TREC 2009 Entity Track. In: TREC (2009)
4. Bast, H., Bäurle, F., Buchhold, B., Haußmann, E.: Semantic full-text search with broccoli. In: SIGIR, pp. 1265–1266. ACM (2014)
5. Bast, H., Buchhold, B.: An index for efficient semantic full-text search. In: CIKM (2013)
6. Bast, H., Buchhold, B., Haussmann, E.: Semantic search on text and knowledge bases. Foundations and Trends in Information Retrieval **10**(2-3), 119–271 (2016). DOI 10.1561/1500000032. URL `http://dx.doi.org/10.1561/1500000032`
7. Bast, H., Chitea, A., Suchanek, F.M., Weber, I.: Ester: efficient search on text, entities, and relations. In: SIGIR, pp. 671–678 (2007)
8. Bast, H., Haussmann, E.: Open information extraction via contextual sentence decomposition. In: ICSC (2013)
9. Bhagdev, R., Chapman, S., Ciravegna, F., Lanfranchi, V., Petrelli, D.: Hybrid search: Effectively combining keywords and semantic searches. In: ESWC, pp. 554–568 (2008)
10. Blanco, R., Halpin, H., Herzig, D.M., Mika, P., Pound, J., Thompson, H.S., Duc, T.T.: Entity search evaluation over structured web data. In: SIGIR Workshop on Entity-Oriented Search (JIWES) (2011)
11. Bollacker, K.D., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: SIGMOD, pp. 1247–1250 (2008)
12. Bron, M., Balog, K., de Rijke, M.: Ranking related entities: components and analyses. In: CIKM, pp. 1079–1088 (2010)
13. Campinas, S., Ceccarelli, D., Perry, T.E., Delbru, R., Balog, K., Tummarello, G.: The sindice-2011 dataset for entity-oriented search in the web of data. In: Workshop on Entity-Oriented Search (EOS), pp. 26–32 (2011)
14. Cimiano, P., Lopez, V., Unger, C., Cabrio, E., Ngomo, A.C.N., Walter, S.: Multilingual question answering over linked data (QALD-3): Lab overview. In: CLEF, pp. 321–332 (2013)
15. Dang, H.T., Kelly, D., Lin, J.J.: Overview of the TREC 2007 Question Answering Track. In: TREC (2007)
16. Giunchiglia, F., Kharkevich, U., Zaihrayeu, I.: Concept search. In: ESWC, pp. 429–444 (2009)
17. Halpin, H., Herzig, D.M., Mika, P., Blanco, R., Pound, J., Thompson, H.S., Tran, D.T.: Evaluating ad-hoc object retrieval. In: Workshop on Evaluation of Semantic Technologies (WEST) (2010)
18. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web **6**(2), 167–195 (2015)
19. Lopez, V., Unger, C., Cimiano, P., Motta, E.: Evaluating question answering over linked data. J. Web Sem. **21**, 3–13 (2013)
20. Mihalcea, R., Csomai, A.: Wikify! Linking documents to encyclopedic knowledge. In: CIKM, pp. 233–242 (2007)
21. Sanderson, M.: Test collection based evaluation of information retrieval systems. Foundations and Trends in Information Retrieval **4**(4), 247–375 (2010)
22. Tablan, V., Bontcheva, K., Roberts, I., Cunningham, H.: Mímir: An open-source semantic search framework for interactive information seeking and discovery. J. Web Sem. **30**, 52–68 (2015)
23. Tran, T., Mika, P., Wang, H., Grobelnik, M.: SemSearch'11: the 4th Semantic Search Workshop. In: WWW (Companion Volume) (2011)
24. Unger, C., Forascu, C., López, V., Ngomo, A.N., Cabrio, E., Cimiano, P., Walter, S.: Question answering over linked data (QALD-4). In: Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014., pp. 1172–1180 (2014)
25. Unger, C., Forascu, C., López, V., Ngomo, A.N., Cabrio, E., Cimiano, P., Walter, S.: Question answering over linked data (QALD-5). In: Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015. (2015)
26. Wang, H., Liu, Q., Penin, T., Fu, L., Zhang, L., Tran, T., Yu, Y., Pan, Y.: Semplore: A scalable IR approach to search the web of data. J. Web Sem. **7**(3), 177–188 (2009)