HAWK – Hybrid Question Answering Using Linked Data

Ricardo Usbeck[1](✉), Axel-Cyrille Ngonga Ngomo[1], Lorenz Bühmann[1], and Christina Unger[2]

[1] University of Leipzig, Leipzig, Germany
{usbeck,ngonga}@informatik.uni-leipzig.de
[2] University of Bielefeld, Bielefeld, Germany
cunger@cit-ec.uni-bielefeld.de

Abstract. The decentral architecture behind the Web has led to pieces of information being distributed across data sources with varying structure. Hence, answering complex questions often requires combining information from structured and unstructured data sources. We present HAWK, a novel entity search approach for Hybrid Question Answering based on combining Linked Data and textual data. The approach uses predicate-argument representations of questions to derive equivalent combinations of SPARQL query fragments and text queries. These are executed so as to integrate the results of the text queries into SPARQL and thus generate a formal interpretation of the query. We present a thorough evaluation of the framework, including an analysis of the influence of entity annotation tools on the generation process of the hybrid queries and a study of the overall accuracy of the system. Our results show that HAWK achieves 0.68 respectively 0.61 F-measure within the training respectively test phases on the Question Answering over Linked Data (QALD-4) hybrid query benchmark.

1 Introduction

Recent advances in question answering (QA) over Linked Data provide end users with more and more sophisticated tools for querying linked data by allowing users to express their information need in natural language [17,19,20]. This allows access to the wealth of structured data available on the Semantic Web also to non-experts. However, a lot of information is still available only in textual form, both on the Document Web and in the form of labels and abstracts in Linked Data sources [9]. Therefore, a considerable number of questions can only be answered by using hybrid question answering approaches, which can find and combine information stored in both structured and textual data sources [22].

In this paper, we present HAWK, the (to best of our knowledge) first full-fledged hybrid QA framework for entity search over Linked Data and textual data. Given a textual input query q, HAWK implements an 8-step pipeline, which comprises (1) part-of-speech tagging, (2) detecting entities in q, (3) dependency parsing and (4) applying linguistic pruning heuristics for an in-depth analysis of

© Springer International Publishing Switzerland 2015
F. Gandon et al. (Eds.): ESWC 2015, LNCS 9088, pp. 353–368, 2015.
DOI: 10.1007/978-3-319-18818-8_22

the natural language input. The results of these first four steps is a predicate-argument graph annotated with resources from the Linked Data Web. HAWK then (5) assign semantic meaning to nodes and (6) generates basic triple patterns for each component of the input query with respect to a multitude of features. This deductive linking of triples results in a set of SPARQL queries containing text operators as well as triple patterns. In order to reduce operational costs, (7) HAWK discards queries using several rules, e.g., by discarding not connected query graphs. Finally, (8) queries are ranked using extensible feature vectors and cosine similarity.

Our main contributions can be summarized as follows:

- We present the first QA framework tackling hybrid question answering;
- HAWK analyses input queries based on predicate-argument trees to deeply understand and match semantic resources;
- Our framework is generic as it does not rely on templates. It is thus inherently able to cover a wide variety of natural language questions.
- The modular architecture of HAWK allows simple exchanging of pipeline parts to enhance testing and deployment;
- Our evaluation suggests that HAWK is able to achieve F-measures of 0.61 on rather small training datasets.

The rest of the paper is structured as follows: Afterwards, our methodology is explained in detail in Sect. 2. HAWK's performance and the influence of entity annotation systems is evaluated in Sect. 3. Section 4 discusses related work. Finally, we conclude in Sect. 5. Additional information can be found at our project home page http://aksw.org/Projects/HAWK.html.

2 Method

In the following, we describe the architecture and methodology of HAWK. We explain our approach by using the following running example: `Which recipients of the Victoria Cross died in the Battle of Arnhem?` While this question cannot be answered by using solely DBpedia or Wikipedia abstracts, combining knowledge from DBpedia and Wikipedia abstracts allows deriving an answer to this question. More specifically, DBpedia allows to retrieve all recipients of the Victoria Cross using the triple pattern `?uri dbo:award dbr:Victoria_Cross.`

In order to find out whether the returned resources died in the Battle of Arnhem, the free text abstract of those resources need to be checked. For example, the abstract for John Hollington Grayburn contains the following information: 'he went into action in the Battle of Arnhem [...] but was killed after standing up in full view of a German tank'.

Figure 1 gives an overview of the architecture of HAWK. In the following we describe the depicted steps in more detail.

2.1 POS-Tagging, Segmentation

A large number of frameworks have been developed for these purposes over the last years. We rely on *clearNLP* [3] which is based on transition-based dependency

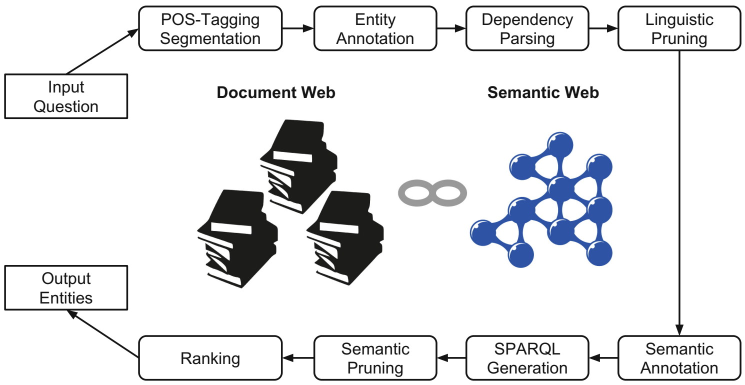

Fig. 1. Architectural overview of HAWK.

parsing and sophisticated segmentation algorithm. Regarding our running example the following POS-tags are generated: `Which(WDT) recipients(NNS) of(IN) the(DT) Victoria(NNP) Cross(NNP) died(VBN) in(IN) the(DT) Battle(NNP) of(IN) Arnhem(NNP)?(PUNCT)`.

2.2 Entity Annotation

HAWK identifies named entities and tries to link them to semantic entities from the underlying knowledge base, in our case DBpedia 3.9, via well-established entity annotation tools, also called entity tagging tools:

- **Wikipedia Miner** [14] is based on different facts like prior probabilities, context relatedness and quality, which are then combined and tuned using a classifier.
- **DBpedia Spotlight** [13] was published in 2011. This tool combines named entity recognition and disambiguation based on DBpedia.
- **TagMe 2** [6] is based on a directory of links, pages and an inlink graph from Wikipedia. The approach recognizes entities by matching terms with Wikipedia link texts and disambiguates the match using the in-link graph and the page dataset.
- **FOX** [18] has been introduced 2014 as an ensemble learning-based approach combining several state-of-the-art named entity recognition approaches. The FOX framework outperforms the current state of the art entity recognizers and relies on the entity linking tool AGDISTIS [23].

Additionally, we implemented two artificial spotters for evaluation:

- **Union** is a spotter that combines the result sets of the above introduced spotters and returns thus a superset of all spotters.
- **Optimal** will spot all entities from the gold standard to be able to ignore spotting influences in the following steps of the pipeline.

For our running example, an optimal spotter identifies `Victoria_Cross` and `Battle_of_Arnhem` as resources form DBpedia. HAWK annotates the POS-tag `ADD` to it. The influence of the entity annotation module is evaluated in Sect. 3.

2.3 Dependency Parsing

HAWK performs noun phrase detection for semantically meaningful word groups not yet recognized by the entity annotation system also known as chunking. This detection reuses the above mentioned POS-tagger. Input tokens will be combined following manually-crafted linguistic heuristics derived from the benchmark questions, and their POS-tag is changed to `CNN`. Thus, the input is a natural language question (list of keywords) and the output is a list of chunks, see Algorithm 1. HAWK's modular structure allows for an easy exchange of the POS-tagger or dependency parser.

Algorithm 1. Algorithm for combining noun phrases

```
Data: Tokenized question ($list$) with Part-of-Speech-tags (POS-tags)
subsequence = ();
for $t \in [0, |list|]$ do
    token = list.get($t$);
    if $subsequence = \emptyset$ then
        if $pos(t) \in$ (CD|JJ|NN(.)*|RB(.)*) then subsequence.add(token);
    else
        if $t + 1 < |list| \wedge pos(t) \in$ (IN) $\wedge\, pos(t+1) \in$ ((W)?DT) then
            if $subsequence.size() >= 2$ then combine(subsequence);
            subsequence = ();
        else if $pos(t-1) \in$ (NNS) $\wedge\, pos(t) \in$ (NNP(S)?) then
            if $subsequence.size() > 2$ then combine(subsequence);
            subsequence = ();
        else if $!pos(t-1) \in$ (JJ|HYPH) $\wedge\, (pos(t) \in$ (VB|WDT|IN))) then
            if $subsequence.size() > 1$ then combine(subsequence);
            subsequence = ();
        else if $pos(t) \in$ (NN(.)*|RB|CD|CC|JJ|DT|IN|PRP|HYPH|VBN)
        then
            subsequence.add(token)
        else
            subsequence = ();
    end
end
```

Subsequently, in order to capture linguistic and semantic relations, HAWK parses the query using dependency parsing and semantic role labeling [3]. The dependency parser is given the chunked question. The generated predicate-argument tree is directed, acyclic, and all its nodes contain their POS-tags as well as their labels, see Fig. 2.

2.4 Linguistic Pruning

The natural language input can contain tokens that are meaningless for retrieving the target information or even introduce noise in the process. HAWK therefore prunes nodes from the predicate-argument tree based on their POS-tags, e.g., deleting all `DET` nodes, interrogative phrases such as `Give me` or `List`, and auxiliary tokens such as `did`. Algorithm 2 details the algorithm for removing nodes. Figure 3 depicts the predicate-argument tree obtained for our running example after pruning.

Algorithm 2. Algorithm for pruning noisy nodes

```
Data: Dependency-argument tree with Part-of-Speech-tags
Queue queue = [tree.getRoot()];
while $queue! = \emptyset$ do
    node = queue.poll();
    if $pos(node) \in$ (WDT|POS|WP$|PRP$|RB|PRP|DT|IN|PDT) then
        tree.remove(node);
    end
    queue.add(node.getChildren());
end
if $root.label ==$ ("Give") then
    for $childNode \in root.getChildren()$ do
        if $childNode ==$ "me" then tree.remove(childNode);
    end
end
if $root.label \in$ {"List", "Give"} then tree.remove(root);
```

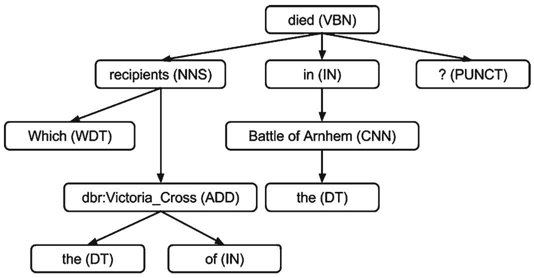

Fig. 2. Predicate-argument tree for the example question 'Which recipients of the Victoria Cross died in the Battle of Arnhem?'

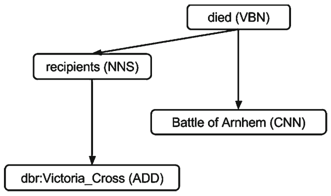

Fig. 3. Tree after pruning. Argument edges are ordered from left to right.

2.5 Semantic Annotation

After linguistic pruning, HAWK annotates each node in the tree with possible concepts from the knowledge base and its underlying ontology. To this end, our framework uses information about possible verbalizations of ontology concepts, based on both `rdfs:label` information from the ontology itself and (if available) verbalization information contained in lexica. In general, such lexica offer a range of lexical variants beyond the labels present in DBpedia. For example, for the property `spouse`, the DBpedia English lexicon[1] provides the noun entries ‘wife’ and ‘husband’ as well as the verb entry ‘to marry’.

HAWK now tries to match each node label to a class or property from the DBpedia ontology using fuzzy string matching. Moreover, HAWK follows intuitions used in [19] to lower the number of annotations avoiding additional computational effort. In particular, we consider the POS-tag of nodes to determine the type of the target reference:

- nouns correspond to object type properties and classes
- verbs correspond to object type properties
- question words (e.g., `who` or `where`) correspond to classes (e.g., `Person` or `Place`).

Afterwards, HAWK ranks properties according to their prominence score in the knowledge base and returns only the top n properties. If the search does not retrieve any annotations, we additionally ask the lemmata of the node label and repeat the above described process to increase recall.

Considering our running example,the nodes `died (VB)` will be annotated with `dbo:deathplace` and `dbo:deathdate` and the node `recipients (NNS)` with `dbo:award`. After this step, either a node is annotated with a reference from the knowledge base or it will be lead to a full-text lookup to be resolved to a knowledge base resource as explained in the following section.

2.6 Generating SPARQL Queries

The core of HAWK is the generation of SPARQL queries from annotated and pruned predicate-argument trees. It uses an Apache Jena FUSEKI[2] server, which implements the full-text search predicate `text:query` on a-priori defined literals over configured predicates. Especially, the following predicates were indexed as they yield a high information content with respect to DBpedia 3.9:

- `dbpedia:abstract` for general interest information about a resource not modelled appropriately in the knowledge base
- `rdfs:label` to match resources not found by the entity annotation system
- `dbpedia:redirect` to identify common synonyms, e.g., ‘first man in space’ pointing to http://dbpedia.org/resource/Yuri_Gagarin
- `dc:subject` for linking top-level categories like ‘assassin’ to resources like http://dbpedia.org/resource/James_Earl_Ray.

Currently, HAWK resolves full-text information either by using exact matches of node labels or fuzzy matches on each non-stopword token of a label; Table 1 depicts the two possibilities for the running example.

[1] https://github.com/cunger/lemon.dbpedia.
[2] http://jena.apache.org/documentation/serving_data/.

Table 1. Examples for full-text query types.

Query type	Query syntax	Node label
Exact	`?var text:query ('Battle of Arnhem')`	Battle of Arnhem
Fuzzy	`?var text:query ('Battle~1 AND Arnhem~1')`	Battle of Arnhem

Table 2. Generated triple patterns for running example.

Node type	Query fragment
CNN	`?proj text:query ('Battle of Arnhem')`
	`?const text:query ('Battle of Arnhem')`
Verb	`?proj dbo:deathPlace ?const`
	`?const dbo:deathPlace ?proj`

To capture the full semantics of an input question, HAWK traverses the predicated-argument tree in a pre-order walk to reflect the empirical observation that (i) related information are situated close to each other in the tree and (ii) information are more restrictive from left to right. This breadth-first search visits each node and generates several *possible triple patterns* based on the number of annotations and the POS-tag itself. That is, for each node a set of SPARQL query patterns is generated following the rules depicted in Table 3 w.r.t. ontology type information, e.g., a variable bound to the class `Place` will not have an outgoing predicate `birthPlace`.

Using this approach allows HAWK to be independent of SPARQL templates and to work on natural language input of any length and complexity. Each pattern contains at least one variable from a pre-defined set of variables, i.e., `?proj` for the resource projection variable, `?const` for resources covering constraints related to the projection variable as well as a variety of variables for predicates to inspect the surrounding of elements in the knowledge base graph. Table 2 shows generated triple patterns for parts of the example query.

During this process, each iteration of the traversal appends the generated patterns to each of the already existing SPARQL queries. This combinatorial effort results in covering every possible SPARQL graph pattern given the predicate-argument tree.

2.7 Semantic Pruning of SPARQL Queries

Producing the n-fold-cross-product of possible pattern combinations generates a huge number of SPARQL queries, most of which are semantically senseless,e.g., a city that has a birth date. To effectively handle this large set of queries and reduce the computational effort, HAWK implements various methods for pruning:

– **#textfilter:** HAWK can safely assume that SPARQL queries containing full-text lookups over more than one variable or containing more than two node labels do not yield semantically senseful information and thus discards such queries.

Table 3. Triple patterns for generating SPARQL queries while traversal.

Node POS-tag and non-empty annotations	Query fragment
VB(.)*	`?proj Annotation ?const.`
VB(.)*	`?const Annotation ?proj.`
VB(.)*	`?const ?proot ?proj.`
NN(.)*|WRB	`?proj Annotation ?const.`
NN(.)*|WRB	`?const Annotation ?proj.`
NN(.)*|WRB	`?proj a Annotation.`
NN(.)*|WRB	`?const a Annotation.`
NN(.)*|WRB	`?const text:query (node label)`
WP	`?const a Annotation.`
WP	`?proj a Annotation.`
In all cases	Add empty triple pattern
Node POS-tag and empty annotations	**Query Fragment**
CNN|NNP(.)*|JJ|CD	`?proj text:query (node label)`
CNN|NNP(.)*|JJ|CD	`?const text:query (node label)`
VB(.)*	`?proj text:query (node label)`
VB(.)*	`?const text:query (node label)`
ADD	`?proj ?pbridge nodeURI.`
ADD	`FILTER (?proj IN (nodeURI))`
ADD	`?proj text:query (node label)`
ADD	`?const text:query (node label)`
NN|NNS	`?proj text:query (node label)`
NN|NNS	`?const text:query (node label)`
In all cases	Add empty triple pattern

- **#unbound triple pattern:** SPARQL queries containing more than one triple pattern of the form `?varx ?vary ?varz` or one such triple pattern and only text searches, lead to a traversal of large parts of the knowledge base graph and high computational effort.
- **Unconnected query graph:** SPARQL query graphs which are not connected from cartesian products are pruned for the sake of runtime and their lack of semantics.
- **Cyclic triple:** Queries containing edges of the form `?s <http://xyz> ?o. ?o <http://xyz> ?s` or `?s <http://xyz> ?o. ?s <http://abc> ?o` are also removed.
- **Missing projection variable:** The before mentioned traversal and SPARQL generation process can produce SPARQL queries without triple patterns containing the projection variable. These queries are also removed from the set of queries.

- **Disjointness:** Also SPARQL queries with triple patterns violating disjointness statements are discarded:
  - `?s a cls . ?s p ?o .` if `cls` and domain of `p` are disjoint
  - `?o a cls . ?s p ?o .` if `cls` and range of `p` are disjoint
  - `?s p1 ?o1 . ?s p2 ?o2 .` if domain of `p1` and `p2` are disjoint
  - `?s1 p1 ?o . ?s2 p2 ?o .` if range of `p1` and `p2` are disjoint
  - `?s p1 ?o . ?s p2 ?o .` if `p1` and `p2` are disjoint.

  Due to lack of explicit disjointness statements in many knowledge bases, we (heuristically) assume that classes and properties that are not related via subsumption hierarchy are disjoint.

Although semantic pruning drastically reduces the amount of queries, it often does not result in only one query. HAWK thus requires a final ranking step before sending the SPARQL query to the target triple store.

2.8 Ranking

HAWK ranks queries using supervised training based on the gold standard answer set from the QALD-4 benchmark. In the *training phase*, all generated queries are run against the underlying SPARQL endpoint. Comparing the results to the gold standard answer set, HAWK stores all queries resulting with the same high F-measure. Afterwards the stored queries are used to calculate an average feature vector comprising simple features mimicking a centroid-based cosine ranking. HAWK's ranking calculation comprises the following components:

- **NR_OF_TERMS** calculates the number of nodes used to form the full-text query part as described in Sect. 2.6.
- **NR_OF_CONSTRAINTS** counts the amount of triple patterns per SPARQL query.
- **NR_OF_TYPES** sums the amount of patterns of the form `?var rdf:type cls`.
- **PREDICATES** generates a vector containing an entry for each predicate used in the SPARQL query.

While running the *test phase* of HAWK, the cosine similarity between each SPARQL query using the above mentioned features and the average feature vector of training queries is calculated. Moreover, HAWK determines the target cardinality x, i.e., `LIMIT x`, of each query using the indicated cardinality of the first seen POS-tag of the input query, e.g., the POS-tag `NNS` demands the plural while `NN` demands the singular case and thus leads to different `x`. The performance of this ranking approach is evaluated in Sect. 3.

3 Evaluation

3.1 Benchmark

We evaluate HAWK against the QALD [21] benchmark. QALD has been used widely to evaluate question answering systems, e.g., TBSL, SINA, FREyA or

QAKiS, which are presented in Sect. 4. In the recent fourth installment of QALD, hybrid questions on structured and unstructured data became a part of the benchmark. To evaluate HAWK, we focus on this hybrid training dataset comprising 25 questions, 17 out of which are entity searches using only DBpedia type information, no aggregation process and require only `SELECT`-queries. The available test dataset comprises only 10 question with 6 entity searches and linguistic structures that are completely different from the training dataset. Before evaluation, we had to curate the benchmark datasets regarding, among others, incorrect grammar, typological errors, duplicate resources in the answer set. The cleaned datasets can be found in our source code repository.[3] Without this correction HAWK's f-measure shrinks to nearly zero for questions containing failures. To the best of our knowledge there is no other published approach on hybrid question answering.

3.2 Influence of the Entity Annotation System

First, we evaluated the influence of the applied entity annotation systems to the overall ability to produce correct answers. Thus, HAWK has been run using DBpedia Spotlight, TagMe, Fox and Wikipedia Miner. Additionally, an optimal entity annotator derived from the gold standard as well as an union of all entity annotation results was analysed.

Our results suggest that HAWK is able to retrieve correct answers with an F-measure of 0.68 using FOX as entity annotation system and assuming an optimal ranking. Furthermore, the optimal ranker is only able to achieve an F-measure of 0.58 since HAWK can cope better with missing annotation results and is tuned towards retrieving full-text information. Against intuition, the Union annotator is the worst annotation system. Merging all annotation results in queries consisting solely of semantic resources eliminating the possibility to match ontology properties and classes to important parts of the query, e.g., matching the word `author` to resource rather than to a property prevents HAWK from generating the correct SPARQL query. Thus, the Union annotator achieves only an F-measure of 0.10.[4]

3.3 Influence of the Ranking Method

Next, evaluating the effectiveness of the feature-based ranking has to include an in-depth analysis of the contribution of each feature to the overall result. Thus, we calculated the power set of the set of features and evaluated each feature group using the F-measure reached by the top n queries. Figures 4 and 5 show the F-measure@N for all query result sets of size N from all 17 questions.

Delving deeper into this analysis, we find:

– Although **NR_OF_TERMS** produces the largest sum of F-measures as a single feature, **NR_OF_CONSTRAINTS** achieves a higher F-measure as

[3] https://github.com/AKSW/hawk/tree/master/resources.
[4] Details on this evaluation can be found in the supplement on our project homepage.

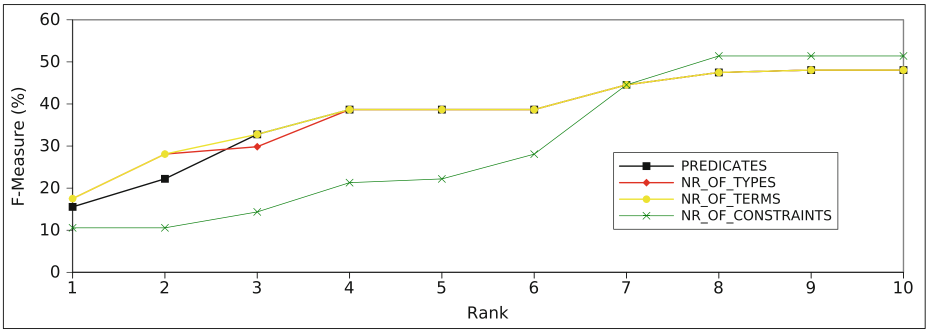

Fig. 4. F-measures on training dataset using $N = [1, \ldots, 10]$ and one feature.

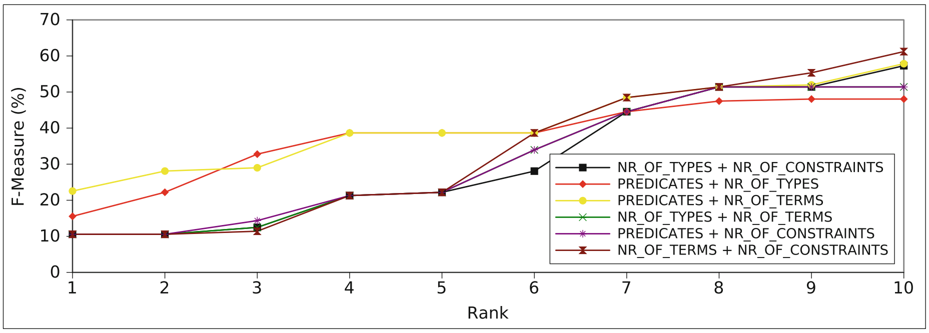

Fig. 5. F-measures on training dataset using $N = [1, \ldots, 10]$ and two features.

soon as $N = 7$ due to the larger number of needed constraints with respect to the query length.
- The highest mass of F-measure reaches the pair **PREDICATES, NR_OF_TERMS** with an F-measure of 0.58 at $N = 10$. However, HAWK is able to achieve a higher F-measure of 0.61 at $N = 10$ using **NR_OF_TERMS, NR_OF_CONSTRAINTS**.
- We only regard the top-10-ranked queries. The correct queries belonged to the top-n queries as shown in Table 4.
- The combination of three or all four features does not lead to an improvement.

HAWK generates up to 15000 SPARQL queries per question containing more than one query generating the correct answer. We consider ranking the resulting SPARQL queries most challenging with respect to the fact that an ideal ranking can lead to F-measures up to 0.72 at $N = 1$.

3.4 Error Analysis

In the following, we analyze error sources in HAWK based on the training queries failing to reach a higher F-measure. Table 4 shows for each entity search question from the training dataset its evaluation results.

Table 4. Micro measures: Precision = 0.70 Recall = 0.85 F-measure = 0.72 at 17 queries from QALD 4 training set. Red indicates inability to generate correct query, Blue indicates missing precision and green missing recall.

ID	Question	F-measure	Precision	Recall
1	Give me the currencies of all G8 countries.	0.0	0.0	0.0
2	In which city was the assassin of Martin Luther King born?	1.0	1.0	1.0
3	Which anti-apartheid activist graduated from the University of South Africa?	1.0	1.0	1.0
5	Which recipients of the Victoria Cross died in the Battle of Arnhem?	0.8	0.67	1.0
6	Where did the first man in space die?	1.0	1.0	1.0
8	Which members of the Wu-Tang Clan took their stage name from a movie?	0.31	0.18	1.0
9	Which writers had influenced the philosopher that refused a Nobel Prize?	0.71	0.56	1.0
11	Who composed the music for the film that depicts the early life of Jane Austin?	0.0	0.0	0.0
14	Which horses did The Long Fellow ride?	1.0	1.0	1.0
15	Of the people that died of radiation in Los Alamos, whose death was an accident?	0.67	1.0	0.5
16	Which buildings owned by the crown overlook the North Sea?	0.25	0.14	1.0
17	Which buildings in art deco style did Shreve, Lamb and Harmon design?	0.5	0.33	1.0
18	Which birds are protected under the National Parks and Wildlife Act?	1.0	1.0	1.0
19	Which country did the first known photographer of snowflakes come from?	1.0	1.0	1.0
20	List all the battles fought by the lover of Cleopatra.	1.0	1.0	1.0
22	Which actress starring in the TV series Friends owns the production company Coquette Productions?	1.0	1.0	1.0
23	Dakar is the capital of which country member of the African Union?	1.0	1.0	1.0

- **Entity Annotation:** Queries 1, 11 and 15 cannot be answered by HAWK due to failing entity annotation. None of the tested annotation tools was able to either find the resources `Jane_T._Austion` nor `G8` or `Los_Alamos`. Without matching entity annotations a full-text search retrieves too many matches for reaching high precision values on limited result set.
- **Missing type information:** some of the resources of the gold standard do not have appropriate type information leading to a high amount of queries that need to be ranked correctly.
- **Query structure:** Queries like 11 or 15 inherit complex query structures leading to a multitude of interpretations while generating the SPARQL query graph.

4 Related Work

Hybrid question answering is related to the fields of hybrid search and question answering over structured data. In the following, we thus give a brief overview of the state of the art in these two areas of research.

First, we present hybrid search approaches which use a combination of structured as well as unstructured data to satisfy an user's information need. Semplore [25] is the first known hybrid search engine by IBM. It combines existing information retrieval index structures and functions to index RDF data as well as textual data. Semplore focuses on scalable algorithms and is evaluated on an early QALD dataset. Bhagdev et al. [1] describe an approach to hybrid search combining keyword searches, Semantic Web inferencing and querying. The proposed K-Search outperforms both keyword search and pure semantic search strategies. Additionally, an user study reveals the acceptance of the Hybrid Search paradigm by end users. A personalized hybrid search implementing a hotel search service as use case is presented in [24]. By combining rule-based personal knowledge inference over subjective data, such as expensive locations, and reasoning, the personalized hybrid search has been proven to return a smaller amount of data thus resulting in more precise answers. Unfortunately, the paper does not present any qualitative evaluation and it lacks source code and test data for reproducibility. All presented approaches fail to answer natural-language questions. Besides keyword-based search queries, some search engines already understand natural language questions. Question answering is more difficult than keyword-based searches since retrieval algorithms need to understand complex grammatical constructs.

Second, we explain several QA approaches in the following. Schlaefer et al. [16] describe *Ephyra*, an open-source question answering system and its extension with factoid and list questions via semantic technologies. Using Wordnet as well as an answer type classifier to combine statistical, fuzzy models and previously developed, manually refined rules. The disadvantage of this system lies in the hand-coded answer type hierarchy. Cimiano et al. [4] developed *ORAKEL* to work on structured knowledge bases. The system is capable of adjusting its natural language interface using a refinement process on unanswered questions. Using F-logic and SPARQL as transformation objects for natural language user queries it fails to make use of Semantic Web technologies such entity disambiguation. Lopez et al. [11] introduce *PowerAqua*, another open source system, which is agnostic of the underlying yet heterogeneous sets of knowledge bases. It detects on-the-fly the needed ontologies to answer a certain question, maps the users query to Semantic Web vocabulary and composes the retrieved (fragment-) information to an answer. However, PowerAqua is outperformed by TBSL (see below) in terms of accuracy w.r.t. the state-of-the-art QALD 3 benchmark. Damljanovic et al. [5] present *FREyA* to tackle ambiguity problems when using natural language interfaces. Many ontologies in the Semantic Web contain hard to map relations, e.g., questions starting with 'How long...' can be disambiguated to a time or a distance. By incorporating user feedback and syntactic analysis FREyA is able to learn the users query formulation preferences increasing the systems question answering precision. Cabrio et al. [2] present a demo of *QAKiS*, an agnostic QA system grounded in ontology-relation matches. The relation matches are based on surface forms extracted from Wikipedia to enforce a wide variety of context matches, e.g., a relation birthplace(person, place) can be explicated by X was born in Y or Y is the birthplace of X. Unfortunately, QAKiS matches only one relation per query and moreover relies on basic heuristics which do not account for the variety of natural language in general. Unger et al. [20] describe *Pythia*, a question answering system based on two steps. First, it uses a domain-independent representation of a query such as verbs, determiners and wh-words. Second, Pythia is based on a domain-dependent, ontology-based interface to transform queries into F-logic. Unfortunately, Pythia does not scale for larger domains since manual mapping of ontology terms via LexInfo is required. Moreover, Unger et al. [19] present a manually curated, template-based approach, dubbed *TBSL*, to match a question against a specific SPARQL query. Combining natural language processing capabilities with Linked Data leads to good benchmark results on the QALD-3 benchmark (see below). TBSL cannot be used to a wider variety of natural language questions due to its limited repertoire of 22 templates. Shekarpour et al. [17] develop *SINA* a keyword and natural language query search engine which is aware of the underlying semantics of a keyword query. The system is based on Hidden Markov Models for choosing the correct dataset to query. *Treo* [8] emphasis the connection between the semantic matching of input queries and the semantic distributions underlying knowledge bases. The tool provides an entity search, a semantic relatedness measure, and a search based on spreading activation. Recently, Peng et al. [15] describe an approach for hybrid QA mapping keywords as well as resource candidates to modified SPARQL queries. Due to its novelty we were not able to compare it to HAWK.

Several industry-driven QA-related projects have emerged over the last years. For example, DeepQA of IBM Watson [7], which was able to win the Jeopardy! challenge against human experts. Further, KAIST's Exobrain[5] project aims to learn from large amounts of data while ensuring a natural interaction with end users. However, it is yet limited to Korean for the moment.

The field HAWK refers to is hybrid question answering for the Semantic Web, i.e., QA based on hybrid data (RDF and textual data). To the best of our knowledge, none of the previous works has addressed this question so far. For further insights please refer to [10,12] which present surveys on existing question answering approaches.

5 Conclusion

In this paper, we presented HAWK, the first hybrid QA system for the Web of Data. We showed that by using a generic approach to generate SPARQL queries out of predicate-argument structures, HAWK is able to achieve up to 0.68 F-measure on the QALD-4 benchmark. Our work on HAWK however also revealed several open research questions, of which the most important lies in

[5] http://exobrain.kr/.

finding the correct ranking approach to map a predicate-argument tree to a possible interpretation. So far, our experiments reveal that the mere finding of the right features for this endeavor remains a challenging problem. We thus aim to apply an automatic feature engineering approach from deep learning in future works to automatically generate the correct ranking function. Moreover, we aim to integrate HAWK in domain-specific information systems where the more specialized context will most probably lead to higher F-measures. Additionally, we will assess the impact of full-text components over regular LD components for QA, partake in the creation of larger benchmarks (we are working on QALD-5) and aim towards multilingual, schema-agnostic queries. Negations within questions and improved ranking will also be considered. Finally, several components of the HAWK pipeline are computationally very complex. Finding more time-efficient algorithms for these steps will be addressed in future works.

Acknowledgments. This work has been supported by the ESF, the Free State of Saxony and the FP7 project GeoKnow (GA No. 318159).

References

1. Bhagdev, R., Chapman, S., Ciravegna, F., Lanfranchi, V., Petrelli, D.: Hybrid search: Effectively combining keywords and semantic searches. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 554–568. Springer, Heidelberg (2008)
2. Cabrio, E., Cojan, J., Gandon, F., Hallili, A.: Querying multilingual DBpedia with QAKiS. In: Cimiano, P., Fernández, M., Lopez, V., Schlobach, S., Völker, J. (eds.) ESWC 2013. LNCS, vol. 7955, pp. 194–198. Springer, Heidelberg (2013)
3. Choi, J.D., Palmer, M.: Getting the most out of transition-based dependency parsing. In: ACL, pp. 687–692 (2011)
4. Cimiano, P., Haase, P., Heizmann, J., Mantel, M., Studer, R.: Towards portable natural language interfaces to knowledge bases - The case of the ORAKEL system. Data Knowl. Eng. **65**(2), 325–354 (2008)
5. Damljanovic, D., Agatonovic, M., Cunningham, H., Bontcheva, K.: Improving habitability of natural language interfaces for querying ontologies with feedback and clarification dialogues. J. Web Semant. **19**, 1–21 (2013)
6. Ferragina, P., Scaiella, U.: Fast and Accurate Annotation of Short Texts with Wikipedia Pages. IEEE software (2012)
7. Ferrucci, D.A., et al.: Building watson: An overview of the DeepQA project. AI Mag. **31**(3), 59–79 (2010)
8. Freitas, A., Oliveira, J.G., Curry, E., O'Riain, S., da Silva, J.C.P.: Treo: combining entity-search, spreading activation and semantic relatedness for querying linked data. In: 1st Workshop on Question Answering over Linked Data (QALD-1) (2011)
9. Gerber, D., Hellmann, S., Bühmann, L., Soru, T., Usbeck, R., Ngonga Ngomo, A.-C.: Real-Time RDF extraction from unstructured data streams. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 135–150. Springer, Heidelberg (2013)

10. Kolomiyets, O., Moens, M.-F.: A survey on question answering technology from an information retrieval perspective. Inf. Sci. **181**(24), 5412–5434 (2011)
11. Lopez, V., Fernández, M., Motta, E., Stieler, N.: PowerAqua: Supporting users in querying and exploring the semantic web. Semant. Web J. **3**, 249–265 (2012)
12. Lopez, V., Uren, V.S., Sabou, M., Motta, E.: Is question answering fit for the semantic web?: A survey. Semant. Web J. **2**(2), 125–155 (2011)
13. Mendes, P.N., Jakob, M., Garcia-Silva, A., Bizer, C.: DBpedia spotlight: Shedding light on the web of documents. In: I-Semantics (2011)
14. Milne, D., Witten, I.H.: Learning to link with wikipedia. In: 17th ACM CIKM (2008)
15. Peng, P., Zou, L., Zhao, D.: On the marriage of SPARQL and keywords. CoRR, abs/1411.6335 (2014)
16. Schlaefer, N., Ko, J., Betteridge, J., Sautter, G., Pathak, M., Nyberg, E.: Semantic Extensions of the Ephyra QA System for TREC (2007)
17. Shekarpour, S., Marx, E., Ngomo, A.-C.N., Auer, S.: Sina: Semantic interpretation of user queries for question answering on interlinked data. J. Web Semant. **30**(3), 39–51 (2014)
18. Speck, R., Ngomo, A.-C.N.: Ensemble learning for named entity recognition. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 519–534. Springer, Heidelberg (2014)
19. Unger, C., Bühmann, L., Lehmann, J., Ngomo, A.N., Gerber, D., Cimiano, P.: Template-based question answering over RDF data. In: 21st WWW Conference, pp. 639–648 (2012)
20. Unger, C., Cimiano, P.: Pythia: Compositional meaning construction for ontology-based question answering on the semantic web. In: Muñoz, R., Montoyo, A., Métais, E. (eds.) NLDB 2011. LNCS, vol. 6716, pp. 153–160. Springer, Heidelberg (2011)
21. Unger, C., Forascu, C., Lopez, V., Ngomo, A.N., Cabrio, E., Cimiano, P., Walter, S.: Question answering over linked data (QALD-4). In: CLEF, pp. 1172–1180 (2014)
22. Usbeck, R.: Combining linked data and statistical information retrieval. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 845–854. Springer, Heidelberg (2014)
23. Usbeck, R., Ngonga Ngomo, A.-C., Röder, M., Gerber, D., Coelho, S.A., Auer, S., Both, A.: AGDISTIS - Graph-based disambiguation of named entities using linked data. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 457–471. Springer, Heidelberg (2014)
24. Yoo, D.: Hybrid query processing for personalized information retrieval on the semantic web. Knowl. Base Syst. **27**, 211–218 (2012)
25. Zhang, L., Liu, Q., Zhang, J., Wang, H., Pan, Y., Yu, Y.: Semplore: An IR approach to scalable hybrid query of semantic web data. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 652–665. Springer, Heidelberg (2007)