Answering Boolean Hybrid Questions with HAWK

Ricardo Usbeck[1], Erik Körner[1], and Axel-Cyrille Ngonga Ngomo[1]

University of Leipzig, Germany
{usbeck,ngonga}@informatik.uni-leipzig.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 an extension for HAWK, a novel search approach for Hybrid Question Answering based on combining Linked Data and textual data. Especially, we introduce our preliminary approach towards answering ASK queries. This approach relies a classification heuristic next to the existing 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. Our results show that these developments lead to HAWK achieving 0.74 F-measure on the ASK queries contained in the Question Answering over Linked Data (QALD-5) hybrid query benchmark assuming an given optimal ranking function.

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 [13,15,16]. This allows access to the wealth of structured data available on the current 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 [8]. 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 [18].

In this paper, we extend HAWK, our hybrid QA system with the capabilities to answer also boolean question, i.e., generate SPARQL queries using `ASK`. Given a textual input query q, HAWK implements the following pipeline, which comprises 1) input segmentation 2) part-of-speech tagging, 3) detecting entities and 4) noun phrases in q, 5) dependency parsing and 6) applying linguistic pruning heuristics for an in-depth analysis of the natural language input. The results of these steps is a predicate-argument graph annotated with resources from the Linked Data Web. HAWK then 7) assigns semantic meaning, i.e., properties and classes, to nodes and 8) 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, 9) HAWK discards queries using several rules, e.g., by discarding not connected query graphs. Here, 10) HAWK applies a simple heuristic to classify `SELECT` and `ASK` queries and 11) modifies the generated SPARQL queries 12) respectively determines the cardinality of the `SELECT` query. Finally, 13) queries are ranked using various ranking methods. The current version of HAWK works on DBpedia as Linked Data source as well as Wikipedia abstracts for full-text information. For the sake of this paper, we will assume the implementation of an optimal ranking function for the generated SPARQL queries since we focus on capturing the semantics of the input question in this paper.

Our main contributions can be summarized as follows: (1) We present HAWK, the first QA framework tackling hybrid question answering. (2) HAWK analyses input questions and classifies them according to a simple heuristic. (3) The modular architecture of HAWK allows simple exchanging of pipeline parts to enhance testing and deployment, e.g., novel ranking or classification functions. (4) Our evaluation suggests that HAWK is able to achieve F-measures of 0.74 on QALD-5 hybrid test set.

The rest of the paper is structured as follows: In Section 2, we briefly describe related work and in Section 3 we detail HAWK's architecture. HAWK's performance and the influence of entity annotation systems is evaluated in Section 4. Finally, we conclude in Section 5. A demo as well as additional information can be found at our project home page `http://aksw.org/Projects/HAWK.html`.

2 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 [21] 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. A personalized hybrid search implementing a hotel search service as use case is presented in [20]. Unfortunately, Yoo's approach [20] 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. Second, we explain several QA approaches for answering natural language questions. Schlaefer et al. [12] describe *Ephyra*, an open-source question answering system and its extension with factoid and list questions via semantic technologies. 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. Lopez et al. [10] 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. Damljanovic et al. [5] present *FREyA* to tackle ambiguity problems when using natural language interfaces. Many ontologies 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. Unger et al. [16] 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. Moreover, Unger et al. [15] present a manually curated, template-based approach, dubbed *TBSL*, to match a question against a specific SPARQL query. Shekarpour et al. [13] develop *SINA* a keyword and natural language query search engine which is aware of the underlying semantics of a keyword query. *Treo* [7] emphasis the connection between the semantic matching of input queries and the semantic distributions underlying knowledge bases. Recently, Peng et al. [11] 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 [6], which was able to win the Jeopardy! challenge against human experts. Further, KAIST's Exobrain[1] 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 more information and related work, please have a look at Usbeck et al. [19].

3 Method

In the following, we will shortly describe the 8-step, modular pipeline of HAWK via this running example: `Did Napoleon's first wife die in France?` For more information please have a look at the full method description [19].

1. Input Segmentation. To be generic with respect to the language of the input question, HAWK uses a modular system that is able of tokenizing even languages without clear separation like Chinese[2]. For English input questions our system relies on the *clearNLP* [3]-framework which provides a.o. a white space tokenizer, POS-tagger and transition-based dependency parsing.

2. Part-of-Speech (POS)-Tagging. HAWK annotates each token with its POS-tag which will be later used to identify possible semantic annotations. POS-tagging

[1] `http://exobrain.kr/`

[2] `https://github.com/clir/clearnlp`

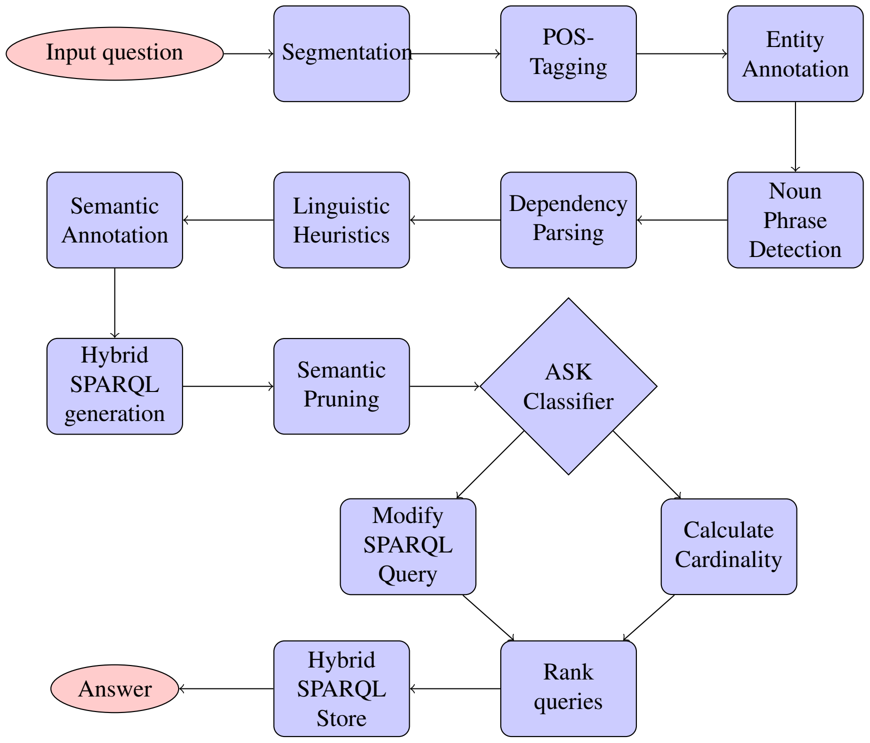

Fig. 1: Pipeline steps of HAWK.

on the running example will result in the following: `Did(VBD) Napoleon(NNP) ’s(POS) first(JJ) wife(NN) die(VB) in(IN) France(NNP) ?(.)`

3. Entity Annotation. Next, our approach identifies named entities and tries to link them to the underlying knowledge base. Most QA challenges, including QALD-5, rely on the DBpedia [9] as source for structured information in the form of Linked Data. For recognizing and linking named entities HAWK’s default annotator is FOX [14], a federated knowledge extraction framework based on Ensemble Learning. An optimal annotator would annotate our running example `Napoleon` with `http://dbpedia.org/resource/Napoleon` and `France` with `http://dbpedia.org/resource/France`.

4. Noun Phrase Detection. HAWK identifies noun phrases, i.e., semantically meaningful word groups, i.e., real-world entities or concepts not captured by the underlying knowledge base, not yet recognized by the entity annotation system, using the result of the POS tagging step. Input tokens are combined following manually-crafted linguistic heuristics based on POS-tag sequences derived from the QALD 5 benchmark questions. Two domain experts implemented the deduced POS-tag sequences and safeguarded the quality of this algorithm w.r.t. the QA pipeline f-measure. Here, the `first wife` would be detected as noun phrase.

5. Dependency Parsing.

To capture linguistic and semantic relations, HAWK parses the query using dependency parsing and semantic role labeling [3]. The dependency parser generates a predicate-argument tree based on the preprocessed question.

6. Linguistic Pruning. Here, HAWK has captured entities from a given knowledge base, noun phrases as well as the semantic structure of the sentence. Still, this structure contains tokens that are meaningless for retrieving the target information or even introduce noise into the process. Thus, HAWK 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`. The linguistically pruned dependency tree with combined noun phrases for our running example would only contain `die` as a root node with two children, namely `first-wife` and `http://dbpedia.org/resource/France`. The node `first-wife` has one further sub-node called `http://dbpedia.org/resource/Napoleon`.

7. Semantic Annotation. Now, the tree structure contains only semantically meaningful (combined) token and entities, i.e, individuals from the underlying knowledge base. To map the remaining token to properties and classes from the target knowledge base and its underlying ontology, our framework uses information about possible verbalizations of ontology concepts and leverages a fuzzy string search. These verbalizations are based on both `rdfs:label`[3] information from the ontology itself and (if available) verbalization information contained in lexica, in our case in the existing DBpedia English lexicon[4]. After this step, either a node is annotated with a class, property or individual from the target knowledge base or it causes a full-text lookup in the targeted Document Web parts. With respect to the running example `die` would be annotated with the properties `dbo:deathPlace` and `dbo:dbo:deathDate`.

8. Generating hybrid SPARQL Queries. Given a (partly) annotated predicate argument, HAWK generates hybrid SPARQL queries. It uses an Apache Jena FUSEKI[5] server, which implements the full-text search predicate `text:query` on a-priori defined literals. Those literals are basically mappings of textual information to a certain individual URI from a target knowledge, i.e., an implicit enrichment of structured knowledge from unstructured data.

Especially, our framework HAWK indexes a-priori the following information per individual: `dbo:abstract`, `rdfs:label`, `dbo:redirect` and `dc:subject` to capture document based information. This information is then retrieved by a special FUSEKI predicate (`text:query`) by using exact matches or fuzzy matches on each non-stopword token of an indexed field.

The generation of SPARQL triple pattern is based on a pre-order walk to reflect the empirical observation that i) related information is situated close to each other in the tree and ii) information is more restrictive from left to right. This breadth-first search visits each node and generates several *possible triple patterns* based on the number of anno-

[3] We assume `dbo` stands for `http://dbpedia.org/ontology`, `dbr` for `http://dbpedia.org/resource/`, `rdfs` for `http://www.w3.org/2000/01/rdf-schema#`, `dc` for `http://purl.org/dc/elements/1.1/` and `text` for `http://jena.apache.org/text#`

[4] `https://github.com/cunger/lemon.dbpedia`

[5] `http://jena.apache.org/documentation/serving_data/`

tations and the POS-tag itself. With this approach HAWK is 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. 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. Amongst others, HAWK generates for the running example the following three hybrid SPARQL queries:

1. `SELECT ?proj {?proj text:query 'first wife'.`
   `?proj dbo:deathPlace dbr:France.`
   `?proj ?pbridge dbr:Napoleon.}`
2. `SELECT ?proj {?proj text:query 'first wife'.`
   `?proj dbo:deathDate dbr:France.`
   `?proj ?pbridge dbr:Napoleon.}`
3. `SELECT ?proj {?proj text:query 'first wife'.`
   `?const pbridge dbr:France.`
   `?proj ?pbridge dbr:Napoleon.}`

9. Semantic Pruning of SPARQL Queries. Covering each possible SPARQL graph pattern with the above algorithm results in a large number of generated SPARQL queries. To effectively handle this large set of queries and reduce the computational effort, HAWK implements various methods for pruning. For example, it assumes that unconnected query graphs, missing projection variables and cyclic SPARQL triple pattern lead to empty or semantically meaningless results. Thus, HAWK discards those queries.

In the running example, the semantic pruning discards query number two from above because it violates the range restriction of the `dbo:deathDate` predicate. Although semantic pruning drastically reduces the amount of queries, it often does not result in only one query. Thus, a ranking of the remaining queries is applied before the best SPARQL query is send to the target triple store.

10. Classification of ASK Queries. To decide whether the user intended a set of entities or a boolean answer as result, HAWK relies on a simple heuristic based on the first word, dubbed indicator word, of the query, see Table 1. We tried using POS tags for the same purposes. However, experiments using POS tags failed due to missing semantics of POS-tags. Furthermore, we acknowledge that classifying questions based on word-level analysis is not language-independent. In the future, we will work on a language independent version of the module leveraging the dependency structure of the input question. Obviously, our running example is classified as `ASK` demanding question.

11. Modify the SPARQL query. After classifying questions and detecting the need for an `ASK` query, HAWK modifies the existing structure, i.e., changes the type of the SPARQL query by replacing the `SELECT` in the query with `ASK`. Furthermore, HAWK

Table 1: Indicator Word for classifying ASK queries in English questions.

Indicator Word (POS-tag)	**Stem form**
Do (VBP), Does (VBZ), Did (VBD)	do
Is (VBZ), Are (VBP), Was (VBD)	be
Have (VBP), Has (VBZ), Had (VBD)	have

skips the cardinality calculation due to `ASK` queries not requiring the `LIMIT` solution modifier.[6]

With respect to our running example, the final query would be:

1.
   ```
   ASK {?proj text:query 'first wife'.
   ?proj dbo:deathPlace dbr:France.
   ?proj ?pbridge dbr:Napoleon}
   ```

12. Cardinality. If HAWK classifies an input question as entity search related rather than demanding a boolean answer, we need to determine the target cardinality x, i.e., modify the solution modifier `LIMIT` x. The number of answers expected for a given query is indicated by cardinality of the first seen POS-tag, e.g., the POS-tag `NNS` demands the plural while `NN` demands the singular case and thus leads to different x. That is, each plural indicating POS-tag will return 10 results by default rather than 1. In the future, we will use a machine learning-based algorithm to learn the correct number of $x \geq 1$.

13. Ranking For the sake of this paper, we assume an optimal ranking algorithm to demonstrate the capabilities of HAWK to identify and generate the correct `ASK` query per input question rather than additionally introducing noise in the process of choosing the correct `ASK` amongst the generated SPARQL queries. To ensure we are able to generate hybrid SPARQL queries capable of answering the benchmark questions, the optimal ranker returns always those hybrid SPARQL queries which lead to a maximum f-measure. Obviously, the optimal ranking can only be used if the answers are know, i.e., HAWK operates on the QALD 5 benchmark datasets containing the gold standard answer set. This ranking functions allows to determine the parts of the hybrid question answering pipeline which do not perform well. An optimal ranking will reveal that the winning SPARQL queries for our running example are:

1.
   ```
   ASK {?proj text:query 'first wife'.
   ?proj dbo:deathPlace dbr:France.
   ?proj ?pbridge dbr:Napoleon},
   ```

2.
   ```
   ASK {?proj text:query ('first wife' AND 'Napoleon') .
   ?proj dbo:deathPlace dbr:France.}.
   ```

[6] `http://www.w3.org/TR/rdf-sparql-query/#ask`

4 Evaluation

The QALD-5 [17] benchmark has a training and a test dataset for question answering containing a subset of hybrid benchmark questions. Moreover, questions depending on Yago ontology[7] types cannot be answered.

The QALD-5 dataset contains 40 training, respectively 10 test questions for hybrid QA. To increase the number of gold standard queries, we did not restrict ourselves to only hybrid, boolean questions but to boolean questions in general, not demanding aggregations, e.g., `FILTER`, and only DBpedia ontology types. Thus, we used 27 questions for the evaluation from the combined dataset following the restrictions given above[8].

Table 2: Results with and without the `ASK`-extension of HAWK.

Question Type	**Global F-measure**	**Global F-measure with skipping**
Hybrid SELECT	0.19	0.27
Hybrid ASK	0.47	0.74
Hybrid SELECT+ASK	0.24	0.35

Table 2 details our results on the combined QALD 5 dataset using an optimal ranker approach. Our simple classification is able to decide in all cases for the correct command method w.r.t. the benchmark data. The *skipping* measure takes into account, iff HAWK does not generate any answer set, i.e., returns 'I do not know the answer'. Overall, the novel implementation of the `ASK`-related modules improves the overall F-measure by more than 10%. A demo of the framework can be found at `http://hawk.aksw.org`.

5 Conclusion

In this paper, we briefly introduced HAWK, a hybrid QA system for the Web of Data, and analysed its performance against the combined QALD 5 dataset using the new `ASK`-query module. We showed that by using a generic approach to generate SPARQL queries from predicate-argument structures, HAWK is able to achieve an F-measure of up to 0.35. Currently, HAWK faces several limitations, such as not capturing the exact semantics due to missing dictionaries (e.g., vice-president), the ability to use `FILTER` and SPARQL aggregation functions (`FILTER (?high > 100)`) or compound questions. Besides, HAWK assumes error-free data sources and user input to focus on the underlying mechanisms that capture the input semantics. Furthermore, the current version of HAWK is based on a frequently modified Java implementation whose

[7] `http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/`

[8] `https://github.com/AKSW/hawk/blob/master/resources/qald-5_test_train.xml`

performance can be experience in the demo. Our work on HAWK, however, revealed several other open research questions, of which the most important lies in finding the correct ranking approach to map a predicate-argument tree to a possible interpretation.

Acknowledgments This work has been supported by the FP7 project GeoKnow (GA No. 318159), the BMWI project SAKE and the Eurostars projects DIESEL and QAMEL.

References

1. R. Bhagdev, S. Chapman, F. Ciravegna, V. Lanfranchi, and D. Petrelli. Hybrid search: Effectively combining keywords and semantic searches. In *ESWC*, pages 554–568. 2008.
2. E. Cabrio, J. Cojan, F. Gandon, and A. Hallili. Querying Multilingual DBpedia with QAKiS. In *ESWC*, pages 194–198, 2013.
3. J. D. Choi and M. Palmer. Getting the most out of transition-based dependency parsing. In *ACL*, pages 687–692, 2011.
4. P. Cimiano, P. Haase, J. Heizmann, M. Mantel, and R. Studer. Towards portable natural language interfaces to knowledge bases - The case of the ORAKEL system. *Data Knowl. Eng.*, pages 325–354, 2008.
5. D. Damljanovic, M. Agatonovic, H. Cunningham, and K. Bontcheva. Improving habitability of natural language interfaces for querying ontologies with feedback and clarification dialogues. *Journal of Web Semantics*, 19:1–21, 2013.
6. D. A. Ferrucci, E. W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefer, and C. A. Welty. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3):59–79, 2010.
7. A. Freitas, J. G. Oliveira, E. Curry, S. O'Riain, and J. C. P. da Silva. 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.
8. D. Gerber, A.-C. Ngonga Ngomo, S. Hellmann, T. Soru, L. Bühmann, and R. Usbeck. Real-time RDF extraction from unstructured data streams. In *ISWC*, 2013.
9. J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 2014.
10. V. Lopez, M. Fernández, E. Motta, and N. Stieler. PowerAqua: Supporting users in querying and exploring the Semantic Web. *Semantic Web Journal*, 3:249–265, 2012.
11. P. Peng, L. Zou, and D. Zhao. On the marriage of SPARQL and keywords. *CoRR*, abs/1411.6335, 2014.
12. N. Schlaefer, J. Ko, J. Betteridge, G. Sautter, M. Pathak, and E. Nyberg. Semantic Extensions of the Ephyra QA System for TREC, 2007. 2007.
13. S. Shekarpour, E. Marx, A.-C. N. Ngomo, and S. Auer. Sina: Semantic interpretation of user queries for question answering on interlinked data. *Journal of Web Semantics*, 2014.
14. R. Speck and A.-C. N. Ngomo. Ensemble learning for named entity recognition. In *ISWC*. 2014.
15. C. Unger, L. Bühmann, J. Lehmann, A. N. Ngomo, D. Gerber, and P. Cimiano. Template-based question answering over RDF data. In *21st WWW conference*, pages 639–648, 2012.
16. C. Unger and P. Cimiano. Pythia: Compositional meaning construction for ontology-based question answering on the Semantic Web. NLDB, pages 153–160, 2011.
17. C. Unger, C. Forascu, V. Lopez, A. N. Ngomo, E. Cabrio, P. Cimiano, and S. Walter. Question answering over linked data (QALD-5). In *CLEF*, 2015.

18. R. Usbeck. Combining Linked Data and Statistical Information Retrieval. In *11th ESWC, PhD Symposium*, 2014.
19. R. Usbeck, A.-C. Ngonga Ngomo, L. Bühmann, and C. Unger. HAWK - Hybrid Question Answering over Linked Data. In *ESWC*, 2015.
20. D. Yoo. Hybrid query processing for personalized information retrieval on the semantic web. *Knowledge Base Systems*, 27:211–218, 2012.
21. L. Zhang, Q. Liu, J. Zhang, H. Wang, Y. Pan, and Y. Yu. Semplore: An IR Approach to Scalable Hybrid Query of Semantic Web Data. In *ISWC*, pages 652–665, 2007.