Hybrid Question Answering over Linked Data

Rawan Bahmid

Thesis submitted to the University of Ottawa

in partial fulfillment of the requirements for the

Master of Computer Science degree

Ottawa-Carleton Institute for Computer Science

School of Information Technology and Engineering

University of Ottawa

© Rawan Bahmid, Ottawa, Canada, 2018

Abstract

The emergence of Linked Data in the form of knowledge graphs in RDF has been one of the most recent evolutions of the Semantic Web. This led to the development of question answering systems based on RDF and SPARQL to allow end users to access and benefit from these knowledge graphs. However, a lot of information on the Web is still unstructured, which restricts the ability of answering questions whose answer does not exist in a knowledge base. To tackle this issue, hybrid question answering has emerged as an important challenge. In fact, hybrid question answering entails the task of question answering by combining both structured (RDF) and unstructured knowledge sources (text) into one answer. This thesis tackles hybrid question answering based on natural language questions. It focuses on the analysis and improvement of an open source system called HAWK, identifies its limitations and provides solutions and recommendations in the form of a generic question-answering pipeline called HAWK_R. Our system mostly uses heuristic methods, patterns and the ontological schema and knowledge base and provides three main additions: question classification, annotation and answer verification and ranking based on query content. Our results show a clear improvement over the original HAWK based on several Question Answering over Linked Data (QALD) competitions. In fact, our methods are not limited to HAWK and can also help increase the performance of other question answering systems.

Acknowledgements

First and foremost, I would like to thank God for giving me the health, strength and patience for accomplishing this thesis. I would also like to convey my deep gratitude and appreciation for my supervisor Dr. Amal Zouaq for her continued support, encouragement, advice, patience, and guidance throughout this thesis. This thesis would not be a success without her assistance.

Many thanks to Dr. Michel Gagnon and Anthony T. Garant (Ecole Polytechnique de Montreal) for the fruitful research discussions.

Thanks to the NLP group meetings and the Tamale seminars for their broad knowledge and cutting-edge research ideas.

In addition, I would like to extend my gratitude and praise to my beloved parents (Khaled & Jameelah) for their love, care, hope and prayers for me. I also would like to thank my sisters and brothers for their faith in me as well my friends who have always encouraged me. Special thanks to my sister Rana for her endless support, care and understanding throughout my master's journey.

I would also like to acknowledge the financial sponsorship of King Abdullah Scholarship Program (KASP) through the Ministry of Higher Education.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

The World Wide Web has become one of the most used tools to seek information. Although Web pages are supposed to be human and machine readable, the HTML language and the diversity of Web pages' formats (textual and multimedia resources) have caused several problems for accessing, extracting and managing information. This makes the Web mostly suitable for human readers but not for machines. To overcome these limitations, Semantic Web technologies have been developed and have evolved into a growing environment of Linked Data sources and applications (Fensel et al., 2011). The Semantic Web is based on automatic data harvesting from unstructured or semi-structured knowledge that is freely available on the internet. This data is stored in knowledge bases made of billions of triples, which are available as Linked Data or as RDF triple stores. RDF triples consist of three components: Subject → Predicate → Object such as *Barack Obama → spouse → Michelle Obama*. One of the main knowledge bases is called DBpedia and it is the RDF representation of data extracted from Wikipedia information boxes (Lehmann et al., 2015). Each Wikipedia page is represented by a DBpedia resource. DBpedia classifies resources using a consistent cross-domain OWL ontology. The ontology contains classes (e.g. persons, places, organizations, and species) and properties (e.g. capital). Knowledge bases such as DBpedia are often queried through SPARQL[1], which is a standard query language for accessing RDF datasets.

The Semantic Web gives an opportunity to implement innovative advanced models of question answering (QA) systems over linked data, which admit full-fledged questions instead of merely a list of keywords. These systems return specific and accurate answers instead of returning a full document that might contain the answer (Yahya et al., 2013). For instance, the answer of "*Who is the wife of Obama?*" using a current Google request would return a list of documents, while question answering systems would return *Michelle Obama* as an answer.

[1] http://www.w3.org/TR/rdf-sparql-query/

1.1 HYBRID QUESTION ANSWERING

The growth of Linked Data gives rise to a growing demand for semantic data exploration. Thus, developing user-friendly interfaces for accessing linked data becomes increasingly important. However, there are two bottlenecks that restrict the interaction between the users and Linked Data. The first issue is that users, whether they are programmers or not, need experience to be able to handle query languages like SPARQL. The second issue is related to the identification and combination of the relevant resources, due to the diversity and high heterogeneity of Linked Data. Question answering over Linked Data is intended to overcome these issues and help users get access to data using natural language questions[2].

Most of the available question answering systems over linked data use only structured data sources to analyze questions and explore answers. However, a lot of information on the Web is still unstructured, which restricts the ability of answering and reaching a huge amount of information (Unger et al., 2014). For instance, to answer the question "*Which anti-apartheid activist graduated from the University of South Africa?*", RDF data can be used to retrieve the resource *dbr:University_of_South_Africa* but it cannot be used to directly identify who is the activist who graduated from the specified university. Therefore, free text search must be used to find articles that mention an "*anti-apartheid activist*", "*University of South Africa*" and "*graduated*". This kind of questions that require both structured and unstructured information and combine such gathered information into one answer are known as *hybrid questions* (Unger et al., 2014) (Unger et al., 2015). This thesis tackles the challenge of hybrid question answering using DBpedia and Wikipedia. More precisely, we present our work in the context of a challenge called the Question Answering over Linked Data (QALD) competition.

1.2 QALD OPEN CHALLENGE

The QALD Open Challenge, which is a part of the QALD competition at the Extended Semantic Web Conference, started in 2011. The challenge encourages designers and researchers to implement question answering systems that translate natural language sentences

[2] https://en.wikipedia.org/wiki/Linked_data
http://linkeddatatools.com/semantic-web-basics
http://linkeddata.org/home
http://www.w3.org/standards/semanticweb/data

into semantic queries. This helps developers to discover the features, strong points and limitations of question answering systems. Each competition proposes specific tasks in order to emphasize various challenges. From QALD-1 until QALD-8, eight different tasks have been proposed. Table 1.1 describes the different tasks, and Table 1.2 illustrates the details of each QALD open challenge. For each task, participants were given specific knowledge base sources, training questions and evaluation methods. DBpedia is the common dataset provided for the challenges. The goal is to evaluate and compare participating systems.

Table 1.1. QALD open challenge tasks

Task	Task title	Description
Task 1	**Question answering over RDF**	Aims at returning proper answers for questions expressed in natural language using a structured knowledge base.
Task 2	**Multilingual question answering over DBpedia**	Aims at extracting correct answers of a natural language question or set of keywords from one or more knowledge bases: English DBpedia, Spanish DBpedia and MusicBrainz in various languages: English, German, Spanish, Italian, French and Dutch.
Task 3	**Ontology lexicalization**	Aims at finding and formatting (lemon lexicon format) English lexicalizations of a varied collection of classes and properties from the DBpedia ontology in a Wikipedia corpus.
Task 4	**Biomedical question answering over interlinked data**	Aims at answering natural language questions by using interlinked data for serving and developing answering systems for life science and medical fields by using three biomedical datasets: SIDER, Diseasome and Drugbank that describe drugs and their side effects, diseases and genetic disorders and FDA-approved active compounds of medication, respectively.
Task 5	**Hybrid question answering over RDF and free text data**	Aims at answering natural language questions that require the combination of structured (RDF) and textual (free text) data to be answered correctly.
Task 6	**Statistical question answering over RDF data cubes**	Aims at using RDF data cube vocabulary to develop QA systems that deal with multi-dimensional and statistical data.
Task 7	**Large-Scale Question answering over RDF**	Aims at building a system that can handle a massive amount of data, enormous number of questions and a parallelization process for speed up.
Task 8	**English question answering over Wikidata**	Aims at building a generic system that can easily adapt to a new data source. Questions primarily formulated for DBpedia have to be answered using Wikidata. Thus, developed systems have to be able to handle different representations and structures of data.

Table 1.2. QALD Open challenge tasks and datasets

Challenge	Datasets	Tasks	Questions	Source
QALD-1	English DBpedia 3.6 MusicBrainz	Task 1	100 training / 100 test	training & training/ test & test
QALD-2	English DBpedia 3.7 MusicBrainz	Task 1	50 training / 154 test	training & training / test & test
QALD-3	English DBpedia 3.8 Spanish DBpedia MusicBrainz	Task 2	250 training / 207 test	training, training & training / test, test & test
		Task 3	40 training / 40 test	training & training / test
QALD-4	English DBpedia 3.9 Free text	Task 2	200 training / 50 test	training / test
		Task 4	25 training / 25 test	training / test
		Task 5	25 training / 10 test	training / test
QALD-5	English DBpedia 2014 Free text	Task 2	300 training / 50 test	training / test
		Task 5	40 training / 10 test	
QALD-6	English DBpedia 2015 Free text	Task 2	350 training / 100 test	training / test
		Task 5	50 training / 25 test	training / test
		Task 6	100 training / 50 test	training / test
QALD-7	English DBpedia 2016-04 Free text	Task 2	215 training / 50 test	training / test
		Task 5	102 training / 50 test	training / test
		Task 7	102 training / 2M test	training / test & test
		Task 8	100 training / 50 test	training / test
QALD-8	English DBpedia 2016-10 Free text	Task 2	219 training / 41 test	Training / test
		Task 5	102 training	training
		Task 8	100 training	training

QA systems can be categorized based on the domain knowledge, answer type and data source. Closed domain QA systems aim at clear and particular domains such as biomedicine in Task 4 and music-related questions using MusicBrainz in Task 1. Open domain QA systems depend on cross-domain data sources and aim at answering generic questions based on DBpedia as a structured data source, or a combination of structured and unstructured data sources.

In the remainder of this thesis, we use *dbr, dbo, rdf, rdfs, owl* and *text* namespaces to describe different Uniform Resource Identifiers (URI) namespaces. *dbr* stands for http://dbpedia.org/resource/, and it designates any entity that can be viewed as an instance. For example, *dbr:Nelson_Mandela* is an instance of a *Person*. *dbo* stands for http://dbpedia.org/ontology/. For instance, *dbr:Nelson_Mandela* is a *dbo:Person* linked to

dbr:Winnie_Mandela through the property *dbo:spouse*. RDF, RDFS and OWL are standard specifications on the Semantic Web and stand for http://www.w3.org/2000/01/rdf, http://www.w3.org/2000/01/rdf-schema and http://www.w3.org/2002/07/owl respectively. The namespace *text* stands for http://jena.apache.org/text#, and it is used to conduct text searches in SPARQL queries.

1.3 Hybrid Question Answering over RDF and Free Text Data

The development of RDF knowledge graphs have made question answering possible on a large scale. However, a lot of information on the Web is still unstructured, which constitutes a real challenge that requires the contribution of both natural language processing and semantic web technologies. For this reason, we focus in this thesis on Task 5 in the QALD competition (see table 1.1), which aims at building systems that support answering questions that require both structured (RDF) and unstructured datasets (free text). An example of a question that seeks both structured and unstructured information to be answered is "*Who is the front man of the band that wrote Coffee & TV?*". Figure 1.1 shows the SPARQL query that contains two triples that reference the DBpedia ontology's properties *dbo:bandMember* and *dbo:musicalArtist*, and one triple of free text. To answer the question, the query must first retrieve the band members of the musical artist associated with the song Coffee & TV from the RDF data using the first two triples, and then check the abstract of the returned URIs to find the frontman of the band. In this case, the abstract of *dbr:Damon_Albarn* contains the following sentence: *He is best known for being the frontman of the Britpop/alternative rock band Blur* [...] (Unger et al., 2016).

```
<question id="335" answertype="resource" aggregation="false" onlydbo="true" hybrid="true">
<string lang =" en"> Who is the front man of the band that wrote Coffee & TV? </string>
<pseudoquery>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?uri
WHERE {
<http://dbpedia.org/page/Coffee_ &_TV >
            dbo: musicalArtist ?x .
            ?x dbo: bandMember ?uri .
            ?uri text :" is"  text :" frontman " .
}
</pseudoquery>
<answers>
<answer> http://dbpedia.org/resource/Damon_Albarn </answer>
</answers>
</question>
```

Figure 1.1. Example of query of Hybrid question - extracted from QALD-6 Training Dataset

Hybrid question answering over RDF and free text data, Task 5, has been offered since QALD-4. All the provided questions for training and test require both structured and unstructured data sources to be answered. The datasets include questions of different types: *WH-questions* (Open questions), *Yes/No questions* (Closed questions) or *imperative-style questions* such as *Give me* and *List* sentences. The answer type can be one or many resources, for which the URI is provided, a string value such as *Die Presse*, a numerical value such as *38* or *2.3*, a date provided in the format YYYY-MM-DD, e.g. *2016-09-23* or either *true* or *false*. Table 1.3 illustrates different types of questions with examples.

Table 1.3. Examples of different answer types - Questions extracted from QALD-7 Training Dataset

Question	**Answer**	**Answer Type**
Which anti-apartheid activist was born in Mvezo?	*http://dbpedia.org/resource/Nelson_Mandela*	Resource
What is the name of the Viennese newspaper founded by the creator of the croissant?	*Die Presse*	String
How old was Steve Job's sister when she first met him?	*25*	Number
When was the composer of the opera Madame Butterfly born?	*1858-12-22*	Date
Did Napoleon's first wife die in France?	*True*	Boolean

1.4 MAIN CONTRIBUTIONS

This thesis focuses on open domain hybrid question answering systems that use English DBpedia and free text as resources. We propose a new system HAWK_R that extends a state of the art system HAWK (Usbeck et al., 2015). There are four main contributions in HAWK_R:

- The first contribution involves the recognition of the question's type. We propose a more *fine-grained classification of question types* using a combination of lexical patterns;

- The second contribution includes a more informed semantic analysis of the question. This includes a verification of the *coherence* of the annotations that are used to semantically annotate the natural language question. This coherence is computed based on the type taxonomy that is extracted from the DBpedia ontology;
- The third contribution aims at filtering candidate answers based on the question type. We propose a *verification of the answer type* based on DBpedia and WordNet;
- The fourth contribution is the implementation of a *ranking mechanism* for queries and answers that is based on a confidence score associated to the content of the queries.

1.5 Organization of the Thesis

The thesis consists of six chapters. Chapter 1 states the main objectives and motivation for the proposed hybrid question answering system. Chapter 2 describes the technologies and methods that are used in question answering over RDF and free text data. After discussing the techniques and architectures of different systems, a generic hybrid question answering system architecture is proposed based on our analysis of the state of the art.

In chapter 3, we describe the architecture of the open source question answering system HAWK (Usbeck et al., 2015), which is used as a baseline. We also describe the datasets used for training and list the evaluation measures. In chapter 4, the limitations and issues of each step of the HAWK pipeline are identified. In chapter 5, we introduce HAWK_R, which proposes several modifications and new solutions to the limits identified in the HAWK modules. Finally, chapter 6 discusses our results and outlines shortcomings and open issues for future work.

CHAPTER 2. STATE OF THE ART

2.1 CHAPTER OVERVIEW

This chapter provides an overview of some question answering systems tools and techniques. Section 2.2 reviews methods used in question answering systems over RDF and question answering systems over RDF and free text data. In section 2.3, we propose a generic question answering system architecture that supports structured and unstructured data based on the synthesis of the main techniques and tools in the reviewed systems.

2.2 QUESTION ANSWERING SYSTEMS

Question answering systems combine several techniques such as natural language processing (NLP) and information retrieval to answer a given question. This section focuses on reviewing techniques behind the QA systems based on structured data in the QALD challenge. The objective is to provide a general overview of QA systems that support RDF data, and illustrate how structured knowledge bases are used to answer questions. These techniques are used as a background for understanding and as a baseline for building more complex systems that support hybrid questions.

We identified four main tasks in the QA process. In the following section, we describe how various QA systems solve them. The tasks are question analysis, query construction, answer verification and answer ranking.

2.2.1 QUESTION ANALYSIS

In this step, the question is analyzed based on syntactic features to find the right tokenization of the question, determine the proper part of speech tags of the tokens, extract the named entities, identify the relations between the tokens and find dependencies of the question components.

2.2.1.1 Tokenization and POS Tagging

Tokenization and part of speech (POS) tagging are one of the essential NLP tasks. Tokenization aims at dividing a sentence into meaningful chunks such as words or tokens.

POS tagging aims at assigning labels to each token based on its part of speech such as NN for nouns and VB for verbs. For example, the sentence "*List all man-made lakes in Australia*" can be split into six tokens and each one is assigned a POS tag: "*List*→VB, *all*→DT, *man-made*→JJ, *lakes*→NN, *in*→IN, *Australia*→NNP". The accuracy of tokenization can affect other tasks such as POS tagging and parsing. For instance, if the same sentence above has a different tokenization then the tokens will have different POS tags: "*man*→NN, -→:, *made*→VBN". Instead of finding only the part of speech as done by a majority of systems (Damljanovic et al., 2010) (Aggarwal & Buitelaar, 2012), some systems build their own taggers to directly detect classes and relations such as Xser (Xu et al., 2014). The Xser system uses machine learning algorithms to tag question's components with (R) for relations, (C) for classes, (V) for variables and (none) for others.

2.2.1.2 Dependency Parsing

The main objective of dependency parsing is to find the syntactic structure of a sentence and the relations between its components. It aims at connecting components to each other according to their grammatical relationships. QA systems use parsers based on different types of grammars. For instance, some systems take into account dependencies between words (Aggarwal & Buitelaar, 2012) (Dima, 2013), whereas others consider dependencies on a phrase level (Hakimov et al., 2015). FREyA (Damljanovic et al., 2011) for example uses the Stanford Parser (Klein & Manning, 2003), which relies on word structure grammars that break down a sentence into its constituent parts and assign tags based on phrasal categories like noun phrase (NP). Systems such as Xser (Xu et al., 2014) and HAWK (Usbeck et al., 2015) use phrase level dependencies.

2.2.1.3 Named Entity Recognition

This important task in question answering aims at identifying tokens that refer to a resource in a knowledge base or a type taxonomy. Some QA systems such as CASIA (He et al., 2013) use named entity (NE) recognition tools such as the Stanford NER tool[3] whereas other systems like HAWK (Usbeck et al., 2015) use entity linking tools such as DBpedia Spotlight[4].

[3] https://nlp.stanford.edu/ner/
[4] https://www.dbpedia-spotlight.org/

Named entity recognition tools identify the tokens that refer to resources of some predefined types (ORGANIZATION, PERSON, etc.). For instance, *Australia* refers to the type LOCATION whereas *Barack Obama* refers to the type PERSON. Entity linking tools recognize NEs, identify the corresponding resources in the KB and perform a disambiguation task. For instance, in the sentence "*List all man-made lakes in Australia*" *Australia* refers to a resource in DBpedia *dbr:Australia*. CASIA uses Stanford NER tool to identify NEs and an N-gram model to map NE to the underlying knowledge base (KB). However, in the N-gram model, the disambiguation is computationally expensive because of the many possible NE candidates (He et al., 2013).

2.2.1.4 Property and Class Mapping

This task aims at mapping the question components that are not named entities to a corresponding property or class in the underlying knowledge base. As an example, the phrase "*lake*" corresponds to the DBpedia class *dbo:Lake*. Different techniques are used to map properties and classes such the use of knowledge base labels and lexicalizations along with string similarity and semantic similarity (Diefenbach et al., 2017). The first step to identify the corresponding class or property of a given word is to look for all the ontology classes / properties in the KB whose label equal to or contains this word. This is done by either using an online triple-store built-in index like in Virtuoso[5] or a local index like Lucene[6]. To deal with string similarity in labels, the distance between the word and the labels from the underlying KB is measured. Lexical databases such as WordNet[7] and Wiktionary[8] and databases for relational lexicalizations such as PATTY[9] (Nakashole et al., 2012), which is a large resource of relational patterns, can be used to deal with semantic similarity. Lexical databases describe all the words' synonyms that can be used to retrieve classes and entities from the KB. Databases specialized in relational lexicalizations are based on collecting and grouping natural language expressions that refer to the same property (relation). In general, QA systems apply several class and property mapping techniques and tools. For instance,

[5] https://dbpedia.org/sparql
[6] https://lucene.apache.org/core/
[7] https://wordnet.princeton.edu
[8] https://www.wiktionary.org
[9] http://www.mpi-inf.mpg.de/yago-naga/patty/

CASIA (He et al., 2013) uses DBpedia index[10], WordNet and PATTY whereas Xser (Xu et al., 2014) uses PATTY only. HAWK (Usbeck et al., 2015) uses a Lucene index combined with Levenshtein distance and stemming for string similarity.

Once the potential candidates have been found based on labels, an ambiguity problem can arise for named entity, class and property mapping. Linking an entity to a resource and mapping a class / property might return multiple possible classes/properties for any word. For instance, the word "*architect*" has two potential mappings *dbo:architect* (a property) and *dbo:Architect* (a class). Question answering systems use different techniques to rank and/or exclude possible resources, classes and properties and to identify the right ones. For example, Xser (Xu et al., 2014) uses a structured perceptron[11] as a machine learning algorithm. This structured perceptron model solves the ambiguity by using string/semantic similarity and popularity of the labels, and a consistency check of the range and domain of properties. Some (semi-automatic) systems allow some user interpretation to perform disambiguation. For instance, FREyA relies on user feedback (Damljanovic et al. 2011) by proposing some resources, classes and properties to the user to choose from.

2.2.2 QUERY CONSTRUCTION

Question answering systems identify the **question type** in order to determine the SPARQL query type SELECT, ASK or COUNT. For instance, Boolean questions are translated into ASK queries, quantity questions use the COUNT keyword and other question types generally are converted into a SELECT query. CASIA (He et al., 2013) is an example of a QA system that uses question words to identify the query type.

Once the question type has been identified, the next task is called **query generation**. During this task, SPARQL queries are built to be sent to a SPARQL endpoint. A query is constructed using templates based on the information extracted from the question analysis phase (POS tags, named entities, classes and properties) or by using machine learning techniques. Some QA systems such as HAWK (Usbeck et al., 2015) and ISOFT (Park et al., 2015) generate SPARQL queries using a collection of predefined query fragments where some slots have to

[10] https://wiki.dbpedia.org/Datasets
[11] http://www.cs.columbia.edu/~mcollins/papers/tagperc.pdf

be filled. Other systems construct queries based on the question analysis. For example, the order of the triples can be decided based on the segmentation of the question, as this is the case in FREyA (Damljanovic et al., 2011) or based on the dependency tree as in QAnswer (Ruseti et al., 2015). Systems such as CASIA (He et al., 2013) use machine learning to extract features from the question to build the query. There are also question answering systems that do not use SPARQL, but navigate through the knowledge base by dereferencing resources/properties, e.g. SemSek (Aggarwal, & Buitelaar, 2012).

2.2.3 ANSWER VERIFICATION

The generated answers are verified by checking their coherence with the expected answer type (EAT) that the question is seeking. EAT is a clue extracted from the question that provides a precise type for the desired answer. The generated answers are compared to the answer type. The EAT can be very generic or more specific. General EAT are based on the classification of the question type. In particular, WH-words are mapped to nouns for instance, *Where* refers to Place or Location, *Who* refers to Person or Organization and *When* refers to date or time. SWIP (Pradel et al., 2013) is an example of a system that uses this technique.

Some systems use mechanisms that obtain a more precise answer type such as ISOFT (Park et al., 2015). ISOFT identifies the core of the question, which is often the noun or verb, (except auxiliary verbs), after the question word that describes the answer type. For example, in the question "*Who is the writer of Jungle Book?*" the general or lexical EAT is "*Who"* which refers to a person or organization, but the specific EAT is "*writer*".

2.2.4 ANSWER RANKING

Some question answering systems rank answers in order to return the most relevant one based on several heuristics such as query confidence score and answer frequency. Some systems rank answers based on the generated SPARQL queries such as SemGraphQA (Beaumont et al., 2015). For example, SemGraphQA assigns scores to query triples based on their content (named entity, class or property). It assigns scores obtained from DBpedia Spotlight to named entities, and scores classes and properties based on the Jaccard distance (Niwattanakul et al., 2013). Then, it uses these scores as confidence scores for the queries. Other systems like

HAWK (Usbeck et.al., 2015) rank answers based on their popularity. The most frequent answer returned by the queries is the top ranked answer.

2.2.5 *SUMMARY*

Based on our analysis of the state of the art of question answering, we can describe and compare the various QA systems by taking into account the most frequent techniques used for the QA task as shown in Table 2.1.

Table 2.1. Comparison of question answering techniques and tools used by the systems for Task 1, Task 2 and Task 5

System	Tokenization & POS tagging	Dependency parsing	Named Entity Recognition	Class/Property Mapping	Disambiguation	Expected Answer Type
FREyA (Damljanovic et al., 2011)		Stanford Parser[12]		√	User Feedback	
SemSek (Aggarwal & Buitelaar, 2012)		Stanford Parser	Lucene[13]	Lucene WordNet[14]	wikiPage-Redirects	
QAKiS (Cabrio et al., 2012)	Stanford CoreNLP[15]	Stanford CoreNLP	Stanford CoreNLP	wiKi-framework	√	
CASIA (He et al., 2014)	Stanford NER[16]	Stanford NER	DBpedia index	PATTY[17] WordNet DBpedia index		
SWIP (Pradel et al., 2013)	?	MaltParser[18]	√	?	User Feedback	√
ISOFT (Park et al., 2015)	ClearNLP[19]	ClearNLP	DBpedia Spotlight[20]	Lucene WordNet	√	
HAWK (Usbeck et al., 2015)	ClearNLP	ClearNLP	DBpedia Spotlight	Lemon[21] DBpedia index		
YodaQA (Baudiš, 2015)	Stanford CoreNLP	Stanford CoreNLP	OpenNLP[22]	WordNet		√
SemGraphQA (Beaumont et al., 2015)		Stanford Parser	DBpedia Spotlight	WordNet DBpedia index		

[12] http://nlp.stanford.edu:8080/parser/
[13] https://lucene.apache.org/core/
[14] https://wordnet.princeton.edu/
[15] http://nlp.stanford.edu:8080/corenlp/
[16] http://nlp.stanford.edu:8080/ner/process
[17] http://www.mpi-inf.mpg.de/yago-naga/patty/
[18] http://www.maltparser.org/
[19] https://clearnlp.wikispaces.com/
[20] https://www.dbpedia-spotlight.org/demo/
[21] https://github.com/cunger/lemon.dbpedia
[22] https://opennlp.apache.org/

(√) symbol indicates that the used technique or tool is not clear from the publication. Empty cell indicates that the technique is not used in that specific system.

2.3 THE QALD COMPETITION

As explained in the introduction, the focus of this thesis is on hybrid question answering in the context of the QALD competition (Task 5). However, we also briefly talk about Task 1 and Task 2, which concentrate on RDF datasets, a component of Task 5.

To give an idea of the performance of state of the art systems in these three tasks, and motivate our work in this thesis, the results of competing systems in Task 1, Task 2 and Task 5 over all QALD challenges are shown in Table 2.2.

In the table, we present the systems and their performances over the specified QALD challenge for each task. *Total* indicates the total number of questions in the dataset. *Processed* indicates the number of questions for which the system was able to provide an answer. *Recall* indicates the number of correct answers with respect to the number of gold standards answers. *Precision* indicates the number of correct answers with respect to the overall number of the system's answers. *F-measure* indicates the harmonic mean of recall and precision. *Macro F-Measure* considers the total number of questions in the datasets whereas *Micro F-measure* considers only the processed questions. The exact formulas are presented in Chapter 3.

Table 2.2. Results for Task 1, Task 2 & Task 5 from QALD-1 to QALD-6 challenges adopted from (Unger et al., 2011), (Unger et al., 2012), (Cabrio et al., 2013), (Unger et al., 2014), (Unger et al., 2015) & (Unger et al., 2016)

Task	QALD	System	Total	Processed	Recall	Precision	Micro F-measure	Macro F-measure
Task 1	QALD-1	FREyA (Damljanovic et al., 2011)	50	43	0.54	0.63		0.58
		PowerAqua (Lopez et al., 2012)		46	0.48	0.52		0.50
	QALD-2	SemSeK (Aggarwal & Buitelaar, 2012)	100	89	0.48	0.44		0.46
		Alexandria		25	0.46	0.43		0.45
		MHE		97	0.40	0.36		0.38
		QAKis (Cabrio et al., 2012)		35	0.37	0.39		0.38
Task 2	QALD-3	squall2sparql (Ferré, 2013)	99	99	0.88	0.93		0.90
		CASIA (He et.al., 2013)		52	0.36	0.35		0.36

		Scalewelis (Guyonvarch & Ferré, 2013)		70	0.33	0.33		0.33
		RTV (Giannone et al., 2013)		55	0.34	0.32		0.33
		Intui2 (Dima, 2013)		99	0.32	0.32		0.32
		SWIP (Pradel et.al., 2013)		21	0.16	0.17		0.17
	QALD-4	Xser (Xu et.al., 2014)	50	40	0.71	0.72		0.72
		gAnswer (Zou et al., 2014)		25	0.37	0.37		0.37
		CASIA (He et.al., 2014)		26	0.40	0.32		0.36
		Intui3 (Dima, 2014)		33	0.25	0.23		0.24
		ISOFT (Park et.al., 2014)		28	0.26	0.21		0.23
		RO_FII		50	0.12	0.12		0.12
	QALD-5	Xser (Xu et.al., 2014)	50	42	0.72	0.74	0.73	0.63
		APEQ		26	0.48	0.40	0.44	0.23
		QAnswer (Ruseti et al., 2015)		37	0.35	0.46	0.40	0.30
		SemGraphQA (Beaumont et.al., 2015)		31	0.32	0.31	0.31	0.20
		YodaQA (Baudiš, 2015)		33	0.25	0.28	0.26	0.18
	QALD-6	CANaLI	100	100	0.89	0.89	0.89	0.89
		UTQA (Pouran-ebn veyseh, 2016)		100	0.69	0.82	0.75	0.75
		KWGAnswer		100	0.59	0.85	0.70	0.70
		NbFramework		163	0.85	0.87	0.86	0.54
		SemGraphQA (Beaumont et.al., 2015)		100	0.25	0.70	0.37	0.37
		UIQA (with manual) (Pouran-ebn veyseh, 2016)		44	0.63	0.54	0.58	0.25
		UIQA (without manual) (Pouran-ebn veyseh, 2016)		36	0.53	0.43	0.48	0.17
Task 5	QALD-5	ISOFT (Park et.al., 2015)	10	3	1.00	0.78	0.87	0.26
		HAWK (Usbeck et.al., 2015)		3	0.33	0.33	0.33	0.10
		YodaQA (Baudiš, 2015)		10	0.10	0.10	0.10	0.10
		SemGraphQA (Beaumont et.al., 2015)		6	0.00	0.20	0.00	0.00
		Xser (Xu et.al., 2014)		3	0.00	0.00	0.00	0.00
	QALD-6	Xser (Xu et.al., 2014)	25	23	0.38	0.43	0.41	0.39
		AskDBpedia		21	0.33	0.51	0.40	0.35
		FirstRun LIMSI		24	0.04	0.61	0.08	0.08

An empty cell indicates that the result is not provided in the published paper. For some QA systems, no corresponding publication could be found even though their results were published in the QALD reports[23].

[23] https://qald.sebastianwalter.org/6/documents/qald-6_results.pdf

For Task 1, FREyA and SemSeK have the highest Macro F-measure in QALD-1 and QALD-2 respectively. For Task 2, Squall2sparql, Xser, and CANaLI performed better than other systems in QALD-3, QALD-4, QALD-5 and QALD-6, respectively. Finally, ISOFT and Xser in QALD-5 and QALD-6 have the best performance among the participating systems for Task 5.

The last two columns of Table 2.3 present, for each task, the average Macro F-measure of all the QA systems and the average Macro F-Measure of the best performing systems (described in the previous paragraph), respectively, over all the datasets that were devoted to the task.

Table 2.3. Median F-measure of the systems

Task	Description	Competition	Median Macro F-measure of all systems	Median Macro F-measure of best performing systems
Task 1	Question answering over RDF	QALD-1 QALD-2	0.46	0.52
Task 2	Multilingual question answering over DBpedia (English DBpedia only)	QALD-3 QALD-4 QALD-5 QALD-6	0.33	0.80
Task 5	Hybrid question answering over RDF and free text data	QALD-5 QALD-6	0.10	0.33

As we can notice from the results shown in the above table, question answering over RDF and hybrid datasets is still an open challenge.

2.4 Generic Question Answering Architecture

Based on our analysis of the state of the art, we propose a generic question answering system architecture that supports structured and unstructured knowledge bases for answering natural language (NL) questions (see Figure 2.1). The generic architecture consists of three main modules: (1) Question Analysis, (2) Information Retrieval and (3) Answer Extraction.

The **Question Analysis** module aims at analyzing the question to detect the question's components that are essential to define the meaning of the question and how to answer it. The analysis involves [1] Semantic annotation, [2] Relation identification and [3] Question type identification.

Semantic annotation includes identifying named entities (NE) from the natural language (NL) question by linking them to their URIs in the knowledge base. Relation identification is used

to recognize the terms in the questions that are classes or properties in the knowledge base. Question type identification aims to define the query template and to classify the expected answers type that the question is seeking. These two tasks are implemented using some specific heuristics and are useful for finding the correct answer candidates.

The second component, **Information Retrieval**, aims at using results of semantic annotation and relation identification to build SPARQL queries. These queries use free text and an RDF knowledge base to extract the results.

In the last component, **Answer Extraction**, the results are filtered based on the expected answer type (EAT). Then these candidate answers are ranked to select the final answer.

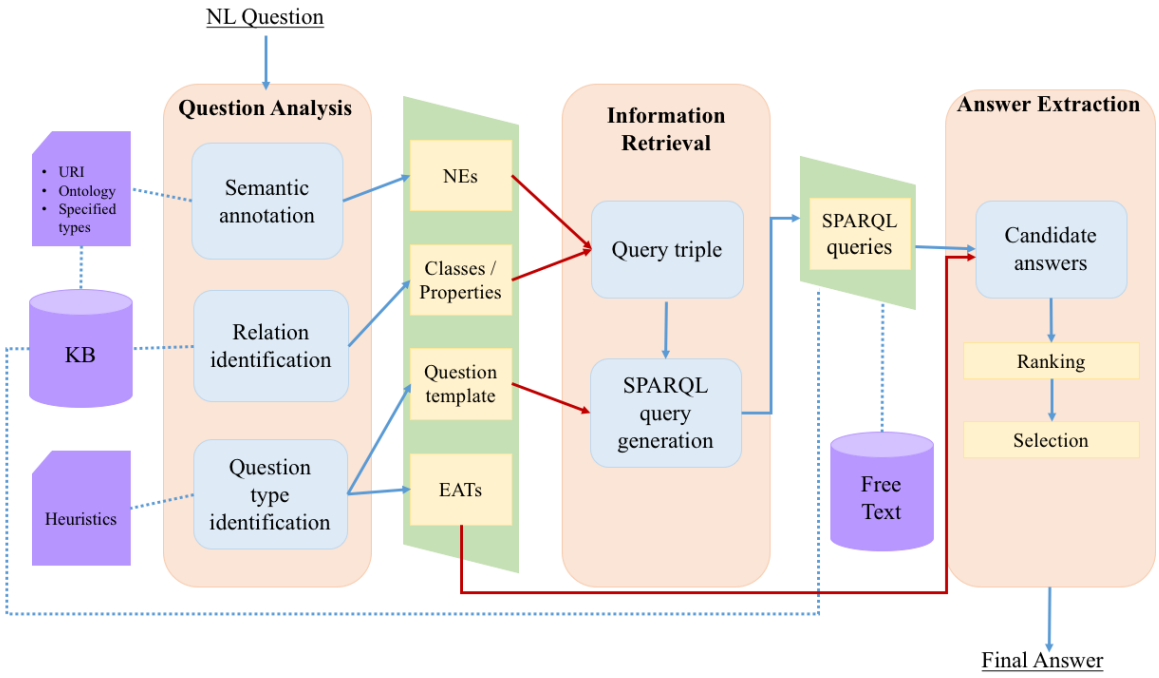


Figure 2.1. Generic system architecture

To implement this general architecture, we decided to use one of the very few open source question answering systems to represent our baseline system. Among the participating systems at the QALD competitions, YodaQA (Baudiš, 2015) and HAWK (Usbeck et al., 2015) are the only open source QA systems for hybrid questions. The main difference between the two systems is that HAWK integrates extracted features from the question into one SPARQL query that uses structured and unstructured KBs to retrieve an answer. In contrast, YodaQA performs

two separate searches in structured and unstructured KBs. We chose HAWK (Usbeck et al., 2015) as it was the top performing system (micro F-Measure) among the two.

2.5 CONCLUSION

After reviewing different systems, it is clear that each system relies on different techniques and strategies to get the maximum benefit of RDF and unstructured data in order to answer questions correctly. However, no system integrates and uses all of these approaches. For instance, YodaQA (Baudiš, 2015) extracts an expected answer type for the question whereas HAWK doesn't. Thus, several strategies derived from the reviewed systems are implemented and tested to improve the performance of HAWK. In the following chapter, we provide a detailed overview of the HAWK question answering system.

CHAPTER 3. HAWK OVERVIEW

HAWK (Usbeck et al., 2015) is an open source question answering system that uses DBpedia[24] as a source of structured RDF data and Wikipedia text as a source of unstructured data. It integrates query segments and text queries to generate a set of SPARQL queries. HAWK consists of 8 modules implemented in Java.

3.1 HAWK ARCHITECTURE AND DESCRIPTION

In this section, the system is explained in detail and illustrated by examples. Figure 3.1 shows the architectural overview of HAWK, which relies on a pipeline of tasks: (1) Question type classification, (2) Segmentation and Part of Speech (POS) tagging, (3) Entity annotation, (4) Phrase combination and Dependency parsing, (5) Linguistic pruning, (6) Semantic annotation, (7) SPARQL generation, (8) Semantic pruning and (9) Ranking.

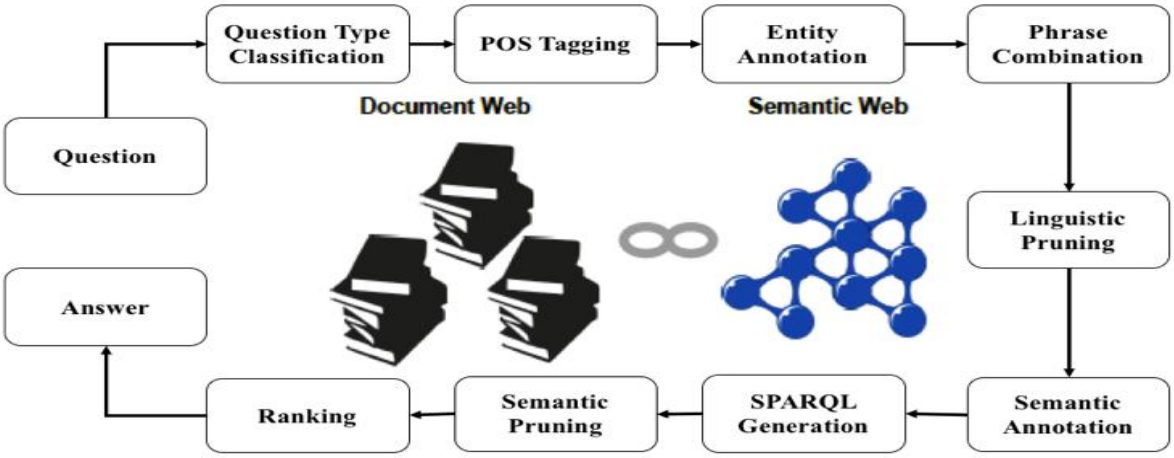


Figure 3.1. Architectural overview of HAWK (Usbeck et al., 2015)

3.1.1 QUESTION TYPE CLASSIFICATION

A question can be a Yes/No question, WH- question or List/Give question as illustrated in the introduction of this thesis. HAWK classifies questions into Boolean and resource questions based on the question word. Questions that start with an auxiliary verb such as "*Are there man-made lakes in Australia that are deeper than 100 meters?*", are classified as Boolean

[24] https://wiki.dbpedia.org/

questions and are based on ASK SPARQL queries. Otherwise, HAWK considers the question as involving a resource and uses a SELECT SPARQL query (`SELECT DISTINCT ?proj WHERE {…}`).

3.1.2 *Segmentation and POS Tagging*

In this step, the question is split into words or tokens and then each token is marked up with a particular part of speech (POS) tag based on both its definition and context. This procedure is done using the online server of the Stanford Core NLP API[25]. Figure 3.2 shows the POS tags of the question "*Which Secretary of State policies lead to the United States' acquisition of Pacific colonies?*".

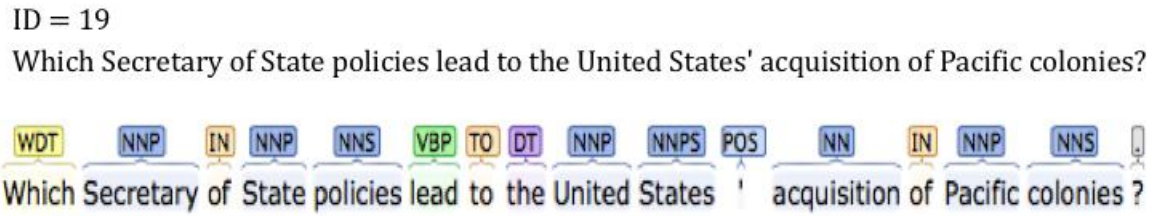


Figure 3.2. Example of POS tagging - Question extracted from QALD-7 Training Dataset, ID=19

3.1.3 *Entity Annotation*

Named Entities (NEs) are real-world objects such as persons or organizations that have pre-defined classifications. In HAWK, some semantic annotators such as DBpedia Spotlight and Fox[26] are used to identify these entities in the question and link them to their identifiers in DBpedia if they exist. After that, each named entity is replaced by its URI in the question, and its POS tag is changed to ADD. To our knowledge the tags(ADD, CNN – see next section)do not have any clear description in HAWK's published papers. In short, these are not acronyms that mean something specific but represent new tags for combined expressions and named entities. Figure 3.3 demonstrates the process of entity annotation in HAWK by replacing "*United States*" by "*dbr:United_States*" and changing its POS tag to ADD.

[25] https://stanfordnlp.github.io/CoreNLP/index.html
[26] http://aksw.org/Projects/FOX.html

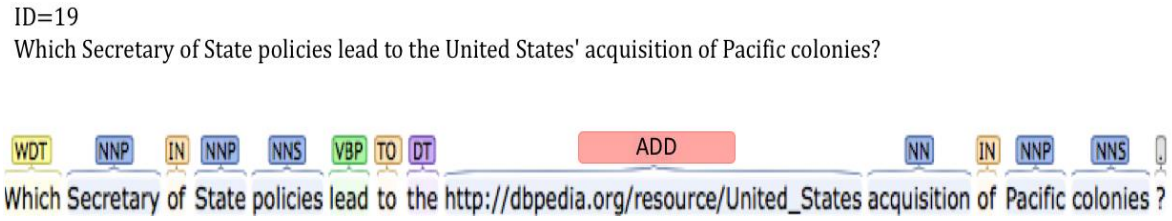


Figure 3.3. Example of entity annotation - Question extracted from QALD-7 Training Dataset, ID=19

3.1.4 Phrase Combination & Dependency Parsing

Phrase Combination. In this step, the system identifies additional noun phrases, which are not tagged by the semantic annotator, and combines them. The combined phrases are assigned a novel POS tag (CNN). Phrases are combined based on five heuristic rules, which rely on the POS tags of question tokens. The heuristic rules are listed in Table 3.1.

Table 3.1. Heuristic rules for combining nominal phrases – POS tags refer to the Penn Treebank Tag set

1. Token [CD|JJ|NN(.)*|RB(.)*]
2. Token [IN] NextToken [(W)?DT|NNP(S)?")]
3. Token [NNP(S)?]
4. Token [VB(.)*|.|WDT] || Token [IN] NextToken [null] || Token [IN] NextToken [DT]
5. Token [NN(.)*|RB|CD|CC|JJ|DT|IN|PRP|HYPH|VBN]

Token refers to the current word that is being checked to be combined. NextToken mentions the word following the Token. (.) indicates the occurrence of 0 to many POS tags following the initial tag. For example, VB(.)* identify the first element as VB followed by sub-types (e.g. VBD or VBN).*

In the question "*Which anti-apartheid activist was born in Mvezo?*" the two tokens "*anti-apartheid*" and "*activist*" are combined and their POS tag becomes CNN, as shown in Figure 3.4.

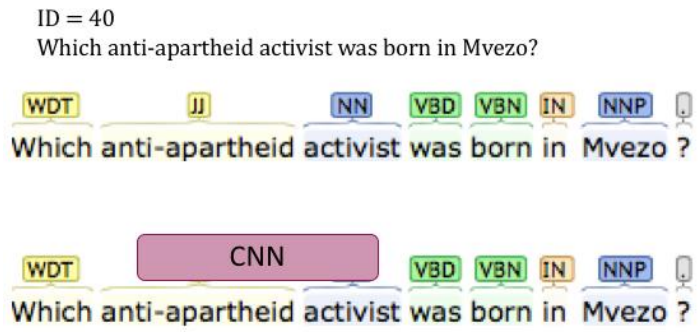


Figure 3.4. Example of phrase combination – Question ID=40 extracted from QALD-7 Training Dataset

Dependency Parsing. After phrase combination, the system uses the Stanford Core NLP API to generate a parsing tree of the question that contains all nodes, their labels and their Part of Speech (POS) tags. Figure 3.5 displays the dependency parse tree of the question:

"*Which anti-apartheid activist was born in Mvezo?*".

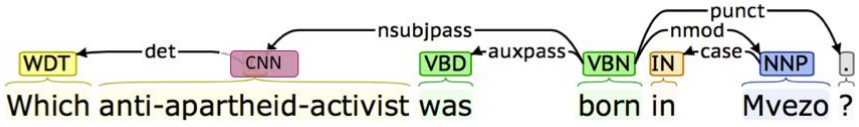


Figure 3.5. Example of dependency parsing – Question ID =40 extracted from QALD-7 Training Dataset

3.1.5 LINGUISTIC PRUNING

After the identification of named entities and combined phrases, the remaining tokens of the question are checked for pruning based on their POS tags such as prepositions and determiners. If the POS tag (Taylor et al., 2003) of a token is "WP", "WP$", "WDT", "WRB", "POS", "WP", "PRP", "RB", "DT", "IN", "PDT" or "." and if a token matches "Give" or "List", or if a token is an auxiliary verb such as "Is", "Was", "Does" and "Has", then that token is pruned. For instance, in the question "*Which anti-apartheid activist was born in Mvezo?*", the tokens "Which $\rightarrow$ *WDT*", "was $\rightarrow$ *VBD*", "in $\rightarrow$ *IN*" and "? $\rightarrow$." are pruned. Thus, the remaining tokens after pruning are "*anti-apartheid activist* $\rightarrow$ *CNN*", "*born* $\rightarrow$ *VBN*" and "*Mvezo* $\rightarrow$ *NNP*".

3.1.6 SEMANTIC ANNOTATION

The purpose of this step is to map nouns and verbs to their corresponding properties and/or classes in the DBpedia ontology. Classes start with capital letters, and can be referred to by nouns only. Properties start with lowercase letters and can be referred to by nouns and verbs. For instance, "*Person* $\rightarrow$ *NN*" refers to *dbo:Person* (a class) and "*reference* $\rightarrow$ *NN" refers* to the datatype property *dbo:reference*. Verbs mostly refer to object properties such as *dbo:starring*.

HAWK uses the DBpedia lexicon[27] to find classes and properties of words (noun / verb). The DBpedia lexicon comprises classes and properties that are extracted from the DBpedia ontology and indexed into a DBpedia class lexicon for classes and a DBpedia property lexicon for properties.

Figure 3.6 demonstrates the process of mapping nouns and verbs to the ontology.

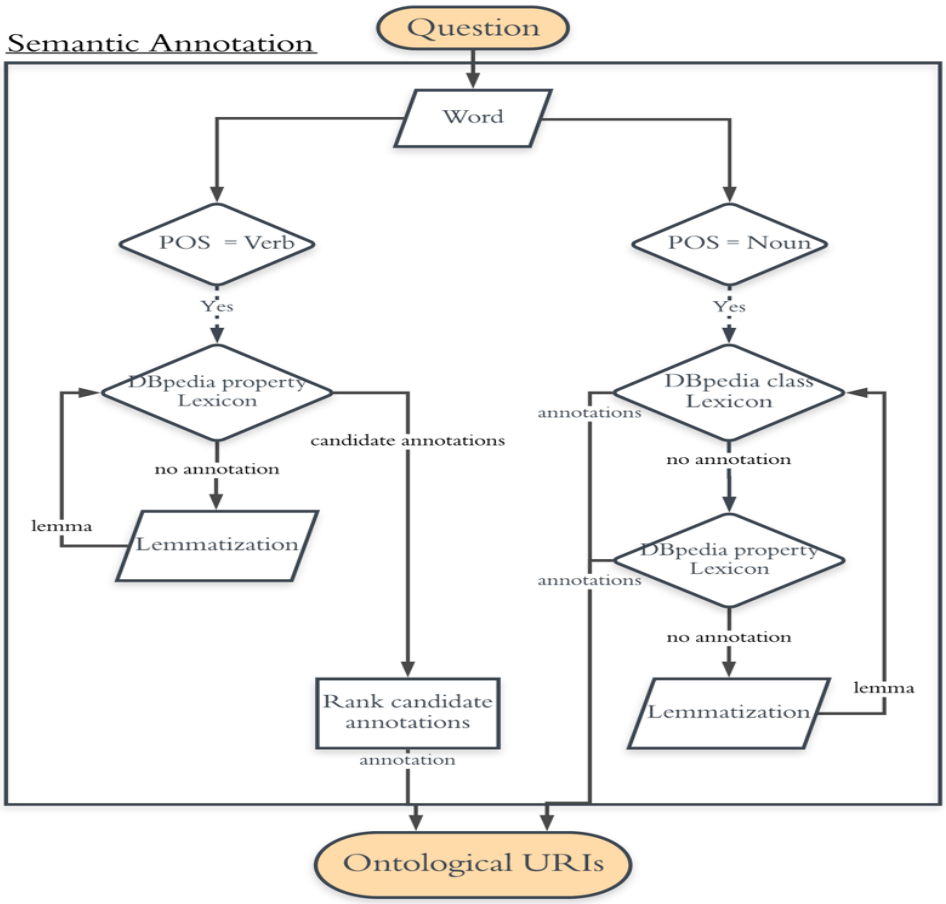


Figure 3.6. Semantic annotation flow chart

For each word in the question, the semantic annotation is performed in the following manner:

Noun: A phrase that is tagged as a noun is looked up in the DBpedia class lexicon to find a proper class using *rdfs:label*. If no match is found, the system checks the DBpedia property lexicon. If we do not find any match using these lexica, we check again using the lemma of

[27] https://github.com/cunger/lemon.dbpedia

the word. If at any stage, we find annotations, these candidate annotations are assigned to the word and we move to check the next word.

Verb: A phrase that is tagged as a verb is looked up in the DBpedia property lexicon to find a proper property using *rdfs:label.* If no match is found, we check again using the lemma of the word. All annotations of a word are ranked based on their frequency of use in DBpedia and the most frequent one is assigned to the verb. For example, the word "*wrote*" can be mapped to two potential properties *dbo:writer* and *dbo:musicalArtist* found in the DBpedia property lexicon. The annotation *dbo:writer* is assigned to "*wrote*" because of its higher frequency.

3.1.7 SPARQL QUERY GENERATION

In this stage, all the previously identified components (words, named entities, combined phrases, properties and classes) are referred as nodes. After analyzing the question and identifying named entities, combined phrases and semantic annotations, SPARQL queries are built by combining query fragments (see Table 3.3). Several queries are generated to cover all potential query forms. In the following sub-sections, we explain the rules of assigning cardinality numbers to queries, the patterns of query fragments and different methods for text search.

3.1.7.1 Cardinality Limit Identification

To restrict the number of returned answers for a query, each none-Boolean SPARQL query is given a limit number 1 or 12, which specifies the maximum number of answers. HAWK arbitrary sets the cardinality to 12 for multiple answer questions. This cardinality number is identified based on the POS tag of the root of the dependency tree for a given question. HAWK Table 3.2 illustrates the rules used by HAWK for setting a limit.

Table 3.2. Rules of cardinality number

1. Root [VB(.)*]		
1. Child [NNS]	=>	Cardinality = 12
2. Child [WP||WRB||ADD||NN||VBZ||IN]	=>	Cardinality = 1
3. Otherwise	=>	Cardinality = 12
2. Root [NNS |NNP(.)*]	=>	Cardinality = 12
3. Otherwise	=>	Cardinality = 1

For example, in Figure 3.7, the root of the question "*Which recipients of the Victoria Cross fought in the Battle of Arnhem?*" is" *fought*" and its POS tag is VBD, and thus the limit is 12. However, the question "*The home of the Mona Lisa is a landmark of which city?*" in Figure 3.8 has limit 1.

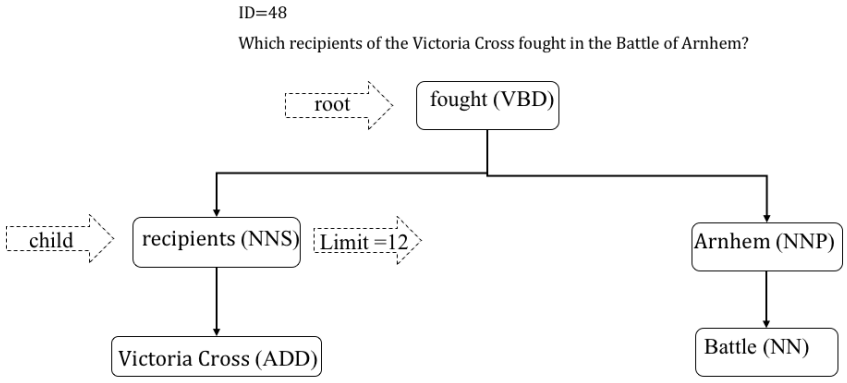


Figure 3.7. Example of cardinality – Question ID=48 extracted from QALD-7 Training Dataset

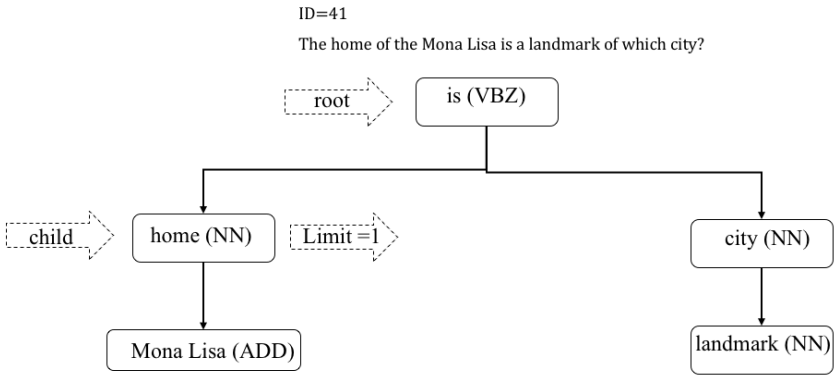


Figure 3.8. Example of cardinality – Question ID=41 extracted from QALD-7 Training Dataset

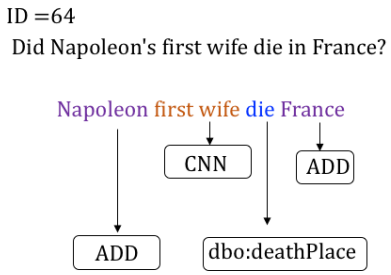
3.1.7.2 Query Fragments

All identified components are linked to query fragments. The query fragments (triples) correspond to patterns that are based on the POS tag of the components or additional tags resulting from the previous modules (such as ADD) and whether or not the components are mapped to a DBpedia resource, class or property. These rules are indicated in Table 3.3. For instance, as shown in the table, a named entity (with tag ADD) is associated to three different query fragments, whereas a combined phrase has only two, because named entities have URIs in DBpedia that the queries must include.

Table 3.3. Query fragments template based on node POS tag.

Nodes mapped to DBpedia Resource/Ontology	**Query fragment**
ADD (Named Entity)	1. ?proj ?pbridge node URI 2. ?proj text:query node label 3. ?const text:query node label
VB(.)*	1. ?proj Annotation ?const 2. ?const Annotation ?proj 3. ?const ?proot ?proj
NN(.)* | WRB	1. ?const Annotation ?proj 2. ?const rdf:type Annotation 3. ?proj rdf:type Annotation 4. ?const text:query node label 5. ?proj text:query node label
WP	1. ?const rdf:type Annotation 2. ?proj rdf:type Annotation
Nodes not mapped to DBpedia Resource/Ontology	**Query fragment**
CNN (Combined Phrase) | NNP(.)* | JJ | CD	1. ?proj text:query node label 2. ?const text:query node label
VB(.)*	1. ?proj text:query node label 2. ?const text:query node label
NN | NNS	1. ?proj text:query node label 2. ?const text:query node label

The following example illustrates the concept of query fragments. In the question "*Did Napoleon's first wife die in France?*", there are four nodes of different types: "*Napoleon* → ADD" and "*France* → ADD" are associated to three query fragments since they are named entities, "*first wife* → CNN" is linked to two query fragments because it is a combined phrase, whereas "*die* → VB" has three query fragments since it is a verb mapped to a property in the DBpedia ontology (see Figure 3.9).



Node	POS tag	Annotation	Query fragment
Napoleon	ADD	dbr:Napoleon	?proj ?pbridge dbr: Napoleon ?proj text:query "Napoleon" ?const text:query "Napoleon"
first wife	CNN	text:first wife	?proj text:query "first wife" ?const text:query "first wife"
die	VB	dbo:deathPlace	?proj dbo:deathPlace ?const ?const dbo:deathPlace ?proj ?const ?proot ?proj
France	ADD	dbr:France	?proj ?pbridge dbr: France ?proj text:query "France" ?const text:query "France"

Figure 3.9. Example of query fragments – Question ID=64 extracted from QALD-7 Training Dataset

3.1.7.3 Text Search

Combined phrases and words that cannot be mapped to any class or property trigger a full-text lookup. This is where the hybrid part of the question answering is taken into account. Combined phrases and words are looked up in Wikipedia pages or DBpedia abstracts (based on the used dataset for each QALD challenge). From instance, in the question "*Did Napoleon's first wife die in France?*", "*first wife*" is a combined phrase that is looked up in Wikipedia pages/DBpedia abstracts.

Textual expressions are looked up in Wikipedia pages/DBpedia abstracts using two different approaches: exact search and fuzzy search.

In exact search, the text is looked up as a block and without any modifications. However, the exact expression might not be found in Wikipedia pages/DBpedia abstracts. Therefore, fuzzy search is used to find words that are very close to the ones we are looking for. Fuzzy search looks for individual words with +/-1 letter in their suffix. Overall, each text search is concretized by two queries: one for exact search and the other for fuzzy search. For instance, in the question" *Did Napoleon's first wife die in France?*" the combined phrase "*first wife*" is looked up as a text in Wikipedia pages/DBpedia abstracts. To perform text search, HAWK

uses Apache Jena[28], a Java framework that integrates SPARQL and full text search via Lucene. Figure 3.10 shows several generated queries. Queries on the left represent exact text search (using the predicate *text:query*) where we look up for "*first wife*" as a block of text whereas queries on the right demonstrate fuzzy search where we look up for Wikipedia pages / DBpedia abstracts that contain "*first*" and "*wife*" as two separate words.

ID =64
Did Napoleon's first wife die in France?

Exact text search	Fuzzy text search

PREFIX text:<http://jena.apache.org/text#>
PREFIX dbr:<http://dbpedia.org/resource/ >
PREFIX dbo:<http://dbpedia.org/ontology/>
ASK WHERE {
?proj ?pbridge dbr: Napoleon
?proj text:query "first wife"
?proj dbo:deathPlace ?const
?proj ?pbridge dbr: France
}
	PREFIX text:<http://jena.apache.org/text#>
	PREFIX dbr:<http://dbpedia.org/resource/ >
	PREFIX dbo:<http://dbpedia.org/ontology/>
	ASK WHERE {
	?proj ?pbridge dbr: Napoleon
	?proj text:query "first~1 AND wife~1"
	?proj dbo:deathPlace ?const
	?proj ?pbridge dbr: France
	}

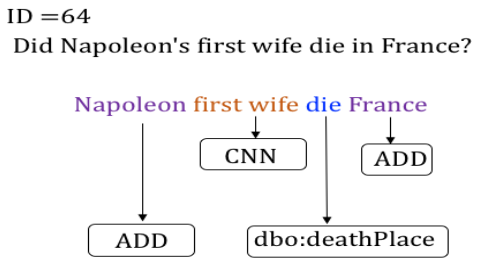
PREFIX text:<http://jena.apache.org/text#>
PREFIX dbr:<http://dbpedia.org/resource/ >
PREFIX dbo:<http://dbpedia.org/ontology/>
ASK WHERE {
?proj text:query "Napoleon"
?const text:query "first wife"
?const dbo:deathPlace ?proj
?proj ?pbridge dbr: France
}
	PREFIX text:<http://jena.apache.org/text#>
	PREFIX dbr:<http://dbpedia.org/resource/ >
	PREFIX dbo:<http://dbpedia.org/ontology/>
	ASK WHERE {
	?proj text:query "Napoleon~1"
	?const text:query "first~1 AND wife~1"
	?const dbo:deathPlace ?proj
	?proj ?pbridge dbr: France
	}

PREFIX text:<http://jena.apache.org/text#>
PREFIX dbr:<http://dbpedia.org/resource/ >
PREFIX dbo:<http://dbpedia.org/ontology/>
ASK WHERE {
?const text:query "Napoleon"
?proj ?pbridge dbr: France
?const ?proot ?proj
}
	PREFIX text:<http://jena.apache.org/text#>
	PREFIX dbr:<http://dbpedia.org/resource/ >
	PREFIX dbo:<http://dbpedia.org/ontology/>
	ASK WHERE {
	?const text:query "Napoleon~1"
	?proj ?pbridge dbr: France
	?const ?proot ?proj
	}

Figure 3.10. Set of generated hybrid queries – Question ID=64 extracted from QALD-7 Training Dataset (~1) this sign is used in fuzzy text search to find exactly the same words except the last letter.

Overall, the final generated SPARQL queries involve the combination of at most one query fragment at a time with each component. This process is repeated to cover all query fragments' combinations and to generate all potential query forms. Moreover, a cardinality limit is assigned to each query. Figure 3.11 shows some SPARQL queries that are generated in HAWK for the question "*Did Napoleon's first wife die in France?*".

[28] https://jena.apache.org/




```
PREFIX text:<http://jena.apache.org/text#>
PREFIX dbr:<http://dbpedia.org/resource/ >
PREFIX dbo:<http://dbpedia.org/ontology/>
ASK WHERE {
?proj      ?pbridge          dbr: Napoleon
?proj      text:query        "first wife"
?proj      dbo:deathPlace    ?const
?proj      ?pbridge          dbr: France
}
```

```
PREFIX text:<http://jena.apache.org/text#>
PREFIX dbr:<http://dbpedia.org/resource/ >
PREFIX dbo:<http://dbpedia.org/ontology/>
ASK WHERE {
?proj      text:query        "Napoleon"
?const     text:query        "first wife"
?const     dbo:deathPlace    ?proj
?proj      ?pbridge          dbr: France
}
```

```
PREFIX text:<http://jena.apache.org/text#>
PREFIX dbr:<http://dbpedia.org/resource/ >
PREFIX dbo:<http://dbpedia.org/ontology/>
ASK WHERE {
?const     text:query        "Napoleon"
?proj      ?pbridge          dbr: France
?const     ?proot            ?proj
}
```

Figure 3.11. Example of SPARQL queries – Question ID=64 extracted from QALD-7 Training Dataset

3.1.8 SEMANTIC PRUNING

Combining all query fragments generates a huge number of SPARQL queries. Therefore, HAWK uses a pruning mechanism to eliminate some queries that do not display consistency. Query consistency is based on two general principles: query correlation and text look up.

Query correlation considers the consistency of the query and ensures that all variables are interlinked. A query that has a variable without any relation with other variables is pruned. Figure 3.12 shows a query of this type on the left, and a correct query on the right. In the first query, there is no relation or linkage between the first two fragments and the third one: the first two fragments use the variable *?proj* whereas the third one uses the variable *?const*. Since the query returns the variable *?proj*, the variable *?const* is useless as it has no impact on the result. Thus this query can be pruned.

ID=25
Which writers converted to Islam?
writers converted published Islam

Pruned	Not Pruned
PREFIX text:<http://jena.apache.org/text#> PREFIX dbr:<http://dbpedia.org/resource/> PREFIX dbo:<http://dbpedia.org/ontology/> SELECT DISTINCT ?proj WHERE { ?proj text:query "converted" ?proj ?pbridge dbr:Islam ?const a dbo:Writer } LIMIT 12	PREFIX text:<http://jena.apache.org/text#> PREFIX dbr:<http://dbpedia.org/resource/> PREFIX dbo:<http://dbpedia.org/ontology/> SELECT DISTINCT ?proj WHERE { ?proj text:query "converted" ?proj ?pbridge dbr:Islam ?proj a dbo:Writer } LIMIT 12

Figure 3.12. Example of query correlation for semantic pruning – Question ID=25 extracted from QALD-7 Training Dataset

Text lookup is used to eliminate queries with more than two *text:query* fragments, which indicate text search. HAWK arbitrarily restricts the number of text query fragments associated with each variable to two. For instance, the query in Figure 3.15 has three *text:query* fragments "*laureate of American lowlife*", "*books*" and "*published*" thus it is pruned.

ID=43
Which books of the laureate of American lowlife are published by Black Sparrow Books?
books laureate of American lowlife published Black Sparrow Books

Pruned	Not Pruned
PREFIX text:<http://jena.apache.org/text#> PREFIX dbr:<http://dbpedia.org/resource/> SELECT DISTINCT ?proj WHERE { ?proj text:query "laureate of American lowlife" ?proj text:query "books" ?proj text:query "published" ?proj ?pbridge dbr:Black_Sparrow_Books } LIMIT 12	PREFIX text:<http://jena.apache.org/text#> PREFIX dbr:<http://dbpedia.org/resource/> SELECT DISTINCT ?proj WHERE { ?proj text:query "laureate of American lowlife" ?proj text:query "books" ?proj ?pbridge dbr:Black_Sparrow_Books } LIMIT 12

Figure 3.13. Example of text look up for semantic pruning – Question ID=43 extracted from QALD-7 Training Dataset

3.1.9 RANKING

After executing all the queries that remain after the pruning stage, each query might return one or several answers that are grouped into what is called an **answer set**. An answer set contains one or many answers based on the result of each query. We adopt here the QALD campaigns evaluation method to assess the performance of HAWK. HAWK uses two ranking techniques to order all the answer sets and returns the top answer set.

The **overlap-based approach** ranks a list of answer sets with respect to their frequency. The frequency of an answer set is the number of queries that return that specific answer set. Thus, it returns the most frequent result of all the SPARQL queries as the final answer. In Figure

3.14, the given question has three answer sets of one element, and one of these answer sets was returned by two queries instead of one in the other cases. Therefore, the answer set [*dbr:James_G._Blaine*] is returned as the final answer set.

ID = 19
Which Secretary of State policies lead to the United States' acquisition of Pacific colonies?

Answer sets	Frequency
http://dbpedia.org/resource/Michael_W._Moore	1
http://dbpedia.org/resource/James_G._Blaine	2
http://dbpedia.org/resource/Council_of_National_Defense	1

Final answer

Figure 3.14. Example of overlap-based ranking – Question ID=19 extracted from QALD-7 Training Dataset

The **optimal ranking** method ranks answers sets with respect to their match to gold standard answers. In fact, this method is only used to ensure that the system is able to correctly answer a given question. However, it cannot be used if the correct answers are not provided, which is the most common scenario of a QA system. For instance, in Figure 3.15, HAWK generates two answer sets for the question "*Which movie established a second carrier for Bill Murray?*". The first answer set is *dbr:Lost_in_Translation_(film)* and its frequency is 3, whereas the second answer set is *dbr:Rushmore_(film)* and it is frequency is 2. Using optimal ranking, the system returns *dbr:Rushmore_(film)* whose F-measure is higher, despite the fact that *dbr:Lost_in_Translation_(film)* has a higher frequency. Thus, the final returned answer is *dbr:Rushmore_(film)*.

ID= 72
Which movie established a second carrier for Bill Murray?
Gold standard answer: http://dbpedia.org/resource/Rushmore_(film)

Answer sets	Frequency	F-measure
http://dbpedia.org/resource/Lost_in_Translation_(film)	3	0
http://dbpedia.org/resource/Rushmore_(film)	2	1

Figure 3.15. Example of optimal ranking – Question ID=72 extracted from QALD-7 Training Dataset

3.2 HAWK Performance in the QALD Competition

This section describes the datasets, metrics and results obtained by HAWK over all the QALD challenges.

3.2.1 Datasets

HAWK officially competed only in the QALD-4 and QALD-5 challenges (see Chapter 1). However, in this thesis, we evaluate HAWK on all QALD[29] datasets that target hybrid question answering and that provide both training and test questions. Each edition of the challenge, from QALD-4 to QALD-8, enriches the previous dataset with new questions, creating an overlap between the datasets of the various competitions (see Table 1.3 in Chapter 1). For instance, the training dataset of the QALD-5 challenge consists of 40 questions with only 18 new questions compared to the previous challenge. We used several datasets to identify HAWK limitations and to train HAWK_R, which include training and test datasets of QALD-4 to QALD-6 in addition to QALD-7 training dataset. To test HAWK_R, we used the QALD-7 test dataset and the QALD-8 training dataset. QALD-7 test dataset consists of 50 hybrid questions, and use DBpedia 2016-04 for structured data and DBpedia abstracts for unstructured data. QALD-8 consists of 102 hybrid questions, and use DBpedia 2016-10 for structured data and Wikipedia pages for unstructured data.

3.2.2 Evaluation Measures

HAWK uses the evaluation measures provided by the QALD competitions to compute recall, precision and F-measure. For each question (q), we compute recall(q) and precision(q). Then, we compute the mean values of recall and precision for the computation of the F-measure. Recall and precision consider both correct and partially correct answers. Answers are considered correct if they match the gold standard answers with no extra answers. However, if the system returns answers that match some gold standard answers or if we have all the correct answers in addition to some extra answers that are not in the gold standard, then the answer is considered partially correct.

[29] https://qald.sebastianwalter.org/

Recall(q) indicates, for a given question q, the number of correct answers with respect to the number of gold standards answers. **Precision(q)** indicates, for a given question q, the number of correct answers with respect to the overall number of the system's answers. **F-measure** indicates the harmonic mean of Recall and Precision. The following formulas are used to compute Recall(q) and Precision(q):

$$Recall(q) = \frac{number\ of\ correct\ system\ answers}{number\ of\ gold\ standard\ answers} \quad (1)$$

$$Precision(q) = \frac{number\ of\ correct\ system\ answers}{number\ of\ system\ answers} \quad (2)$$

The performance measures are computed both globally (Macro) and locally (Micro).

Macro Recall (Mean) indicates the average Recall values for all the questions. **Macro Precision** (Mean) indicates the average Precision values for all the questions. **Macro F-measure** is calculated with respect to all questions of the dataset. The following formulas are used to compute Macro values of Recall, Precision and F-measure:

$$Macro\ Recall = \frac{Recall(q_1) + \ldots\ldots + Recall(q_n)}{total\ number\ of\ questions} \quad (3)$$

$$Macro\ Precision = \frac{Precision(q_1) + \cdots\ldots + Precision(q_n)}{total\ number\ of\ questions} \quad (4)$$

$$Macro\ F\text{-}measure = \frac{2\ X\ Macro\ Recall\ X\ \ Macro\ Precision}{Macro\ Recall + Macro\ Precision} \quad (5)$$

Micro Recall (Mean) indicates the average Recall values for all the processed questions only. **Micro Precision** (Mean) indicates the average Precision values for all the processed questions only. **Micro F- measure** is calculated with respect to only the questions that provide answers. The following formulas are used to compute Micro Recall, Precision and F-measure:

$$Micro\ Recall = \frac{Recall(q_1) + \ldots\ldots + Recall(q_n)}{number\ of processed\ questions} \quad (6)$$

$$Micro\ Precision = \frac{Precision(q_1) + \cdots\ldots + Precision(q_n)}{number\ of\ processed\ questions} \quad (7)$$

$$Micro\ F\text{-}measure = \frac{2\ X\ \ Micro\ Recall\ X\ Micro\ Precision}{Micro\ Recall + Micro\ Precision} \quad (8)$$

Given a set of example questions, Table 3.4 and Table 3.5 illustrate the process of computing the evaluation measures.

Table 3.4. An example of results per question - extracted from the QALD-5 Test dataset

Question	**HAWK Answer**	**Gold Standard Answer**	**Rec. (q)**	**Pre. (q)**	**F-measure**
Where was the "Father of Singapore" born?	dbr:Dan_Rather	dbr:Jamaica	0	0	0
Which Secretary of State was significantly involved in the United States' dominance of the Caribbean?	dbr:James_G._Blaine	dbr:James_G._Blaine	1	1	1
Which German mathematicians were members of the von Braun rocket group?	dbr:Walter_Haeussermann dbr:Oswald_Lange dbr:Walter_Jacobi dbr:Ernst_Steinhoff dbr:Dieter_Grau	dbr:Walter_Haeussermann dbr:Oswald_Lange	1	$\frac{2}{5}$ = 0.4	0.57

Some statistics about questions, Recall, Precision and F-measure of Table 3.4 are shown in Table 3.5. The table consists of 9 columns that can be interpreted as follows:

- *Total # of questions* specifies the number of questions in the dataset.
- # of *processed questions* shows the number of questions for which the system provides an answer.
- # of correctly answered questions specifies how many of the processed questions are answered correctly, which means that the final answer set contains all the required answers.
- # of *partially answered questions* specifies how many of the processed questions have a part of the correct answer; that is, they contain a subset of the answers in the final answer set.
- # of *incorrectly answered questions* specifies how many questions are answered wrongly, not processed or do not generate any SPARQL query.

Table 3.5. Example of the calculation of evaluation metrics for HAWK

Total # of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	**Macro *Recall***	**Macro *Precision***	***Macro F-measure***	***Micro F-measure***
3	2	1	1	2	0.66	0.47	0.56	0.82

3.2.3 HAWK RESULTS

As described in section 3.1.9, HAWK returns the most frequent answer among the set of answers with both overlap-based ranking and optimal ranking. We adopt here the QALD campaigns evaluation method to assess the performance of HAWK using overlap-based and optimal ranking.

Table 3.6 and Table 3.7 recapitulate HAWK's performance over different competitions of the QALD challenge.

Table 3.6. HAWK's performance using overlap-based ranking over different datasets

QALD	***Total #*** of questions	# of ***Processed*** questions	# of ***Correctly*** answered questions	# of ***Partially*** answered questions	# of ***Incorrectly*** answered questions	***Macro Recall*** (Mean)	***Macro Precision*** (Mean)	***Macro F-measure***	***Micro F-measure***
QALD-4 Training Dataset	25	22	0	4	21	0.058	0.090	0.071	0.081
QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
QALD-5 Training Dataset	40	34	0	3	37	0.054	0.042	0.047	0.056
QALD-5 Test dataset	10	9	2	1	7	0.250	0.300	0.272	0.303
QALD-6 Training Dataset	49	41	5	4	40	0.140	0.141	0.144	0.172
QALD-6 Test dataset	24	16	1	0	23	0.042	0.042	0.042	0.063
QALD-7 Training Dataset	102	85	5	7	90	0.093	0.101	0.097	0.123
QALD-7 Test dataset	50	26	2	0	48	0.040	0.040	0.040	0.077
QALD-8 Training Dataset	102	86	6	7	89	0.105	0.112	0.109	0.126

Table 3.7. HAWK's performance using optimal ranking over different datasets

QALD	***Total* # of** questions	# of ***Processed*** questions	# of ***Correctly*** answered	# of ***Partially*** answered	# of ***Incorrectly*** answered	***Macro Recall*** (Mean)	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
QALD-4 Training Dataset	25	22	0	9	16	0.243	0.176	0.204	0.232
QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
QALD-5 Training Dataset	40	34	1	8	31	0.151	0.112	0.128	0.151
QALD-5 Test dataset	10	9	2	2	6	0.350	0.383	0.366	0.406
QALD-6 Training Dataset	49	41	6	9	34	0.256	0.254	0.255	0.305
QALD-6 Test dataset	24	16	2	3	19	0.150	0.179	0.164	0.245
QALD-7 Training Dataset	102	85	10	19	73	0.224	0.205	0.214	0.257
QALD-7 Test dataset	50	26	2	1	47	0.043	0.043	0.043	0.083
QALD-8 Training Dataset	102	86	11	18	73	0.237	0.232	0.234	0.278

In Table 3.6, we can see that the F-measure of HAWK on the QALD-5 Test dataset is 0.27, which is higher than the performance reported in the published paper (Usbeck et al., 2015). This is due to the fact that our results are computed with the latest and improved version of HAWK (2017 at testing time). Table 3.8 shows the published and tested performance of HAWK on QALD-5 Test dataset.

Table 3.8. The performance of different versions of HAWK (QALD-5 Test dataset)

QALD		***Total* #** of questions	# of ***Processed*** questions	# of ***Correctly*** answered questions	# of ***Partially*** answered questions	# of ***Incorrectly*** answered questions	***Macro Recall*** (Mean)	***Macro Precision*** (Mean)	***Macro F-measure***	***Micro F-measure***
QALD-5 Test dataset	HAWK 2015 [30]	10	3	1	0	9	0.10	0.10	0.10	0.33
	HAWK 2017 [31]	10	9	2	1	7	0.25	0.30	0.27	0.30

[30] https://github.com/dice-group/hawk
[31] https://github.com/dice-group/NLIWOD

In this chapter, we explained the various steps of HAWK and illustrated them by examples. Moreover, we described the evaluation metrics provided by the QALD campaigns which are used in this thesis to measure HAWK's performance.

CHAPTER 4. HAWK LIMITATIONS

4.1 CHAPTER OVERVIEW

In this chapter, we analyze HAWK's weaknesses and shortcomings in detail to motivate our proposal in chapter 5. Moreover, the limitations of each step are illustrated by examples to show how these issues are restricting and/or reducing the possibilities of obtaining correct and accurate answers. The tasks (e.g. POS Tagging) that are not related to any identified limitation are omitted.

4.2 QUESTION TYPE CLASSIFICATION

HAWK processes Boolean and resource questions based on the question word. This causes all non-Boolean questions to be considered as resource questions, but this is not always correct. For instance, the question "*When was the composer of the opera Madame Butterfly born?*" requires a date answer (a literal) but HAWK incorrectly classifies it as a resource question. Moreover, this coarse-grained classification is too generic to verify the adequacy of the returned answer for this question. For example, if the question in the example above was classified as a "date question", we could check the returned answers and disregard any non-date answer.

Due to its classification of question types into resource and Boolean questions, HAWK eliminates any query whose result does not fall into these categories. This prevents all literal questions (date, number, string, etc.) from being answered correctly. Figure 4.1 shows an example where the correct answer of a question is a date, but HAWK returns a resource URI as the answer.

ID=12
When was the composer of the opera Madame Butterfly born?

Query	PREFIX text:<http://jena.apache.org/text#>
	PREFIX dbr:<http://dbpedia.org/resource/>
	PREFIX dbo:<http://dbpedia.org/ontology/>
	PREFIX dbp:<http://dbpedia.org/property/>
	SELECT DISTINCT ?proj WHERE {
	?const text:query "opera".
	?const ?pbridge dbr: Madama_Butterfly.
	?const dbp:composer ?proj.
	?proj dbo:birthDate ?const.
	}

HAWK answer	http://dbpedia.org/resource/Giacomo_Puccini
Gold standard answer	1858-12-22

Figure 4.1. Example of question type limitation – Question ID=12 extracted from QALD-7 Training Dataset

4.3 Phrase Combination & Dependency Parsing

Phrase combination is used to combine tokens, that is, multi-words expressions that are parts of the question and that are not recognized as named entities. However, HAWK's combination rules have some issues that restrict the possibility of having meaningful combined phrases. The errors are caused by three word categories: adjectives/adverbs, quantity words and prepositions.

Adjective/Adverb. Adjectives and adverbs are not combined with noun phrases, although they provide additional description that is important for the meaning of the phrase and they help find the proper information in text search. An example is shown in Figure 4.2(a) where "*tallest*" is not combined with "*building*". Thus any building can be returned, and not only the tallest one.

Quantity word. Quantity words such as *many* and *much* help to know that the question is of type numeric, and that the answer should be a number. However, combining these words with other phrases is not relevant in text search because in most cases, quantity words do not exist

in the text. Figure 4.2(b) shows an example of this problem where "*many" and "awards*" are combined forming the phrase "*many awards*".

Preposition. Prepositions have significant importance because they link several portions of the sentence. However, when they are used alone, they do not help in finding the right answer. HAWK combines certain prepositions such as *of* with nominal phrases, but these types of combined expressions are not mapped to a corresponding class or property in the knowledge base. On the example presented in Figure 4.2(c), "*capital of*" is combined and thus "*capital*" fails to be mapped to the class *dbo:Capital* using the current mapping techniques.

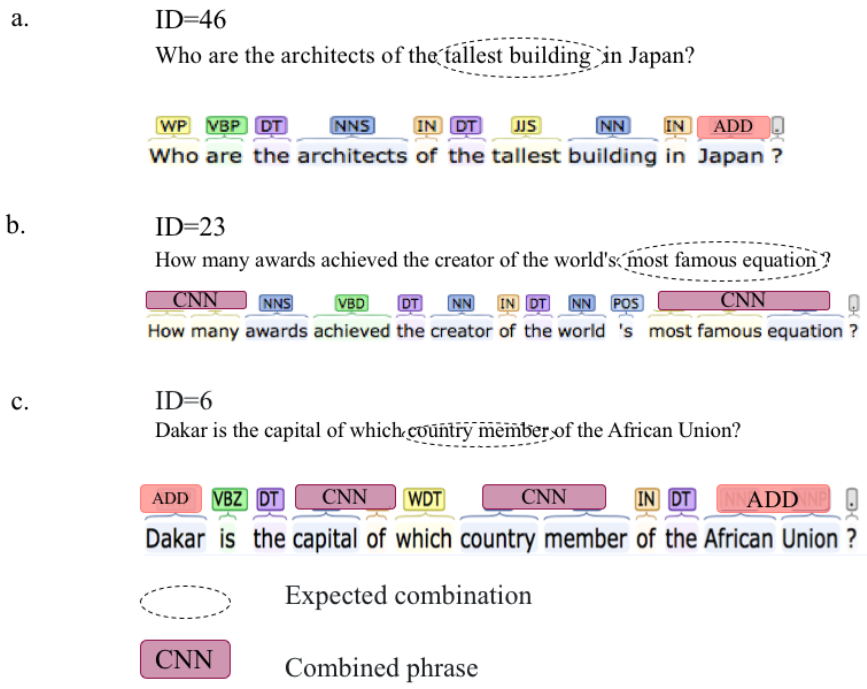


Figure 4.2. Examples of phrase combination – Question ID= 46, 23, 0 extracted from QALD-7 Training Dataset

4.4 Linguistic Pruning

Linguistic pruning is used to prune parts of the question that are not named entities nor combined phrases (e.g. determiners). However, some noisy words are not pruned such as auxiliary verbs and question words although some of them were already used to identify the type of the question and hence are not necessary anymore. These words could lead to incorrect queries and answers. In the example in Figure 4.3, the words "*in*, *that*, *then*" are not pruned and hence they are part of the text search queries in Wikipedia pages/DBpedia abstracts.

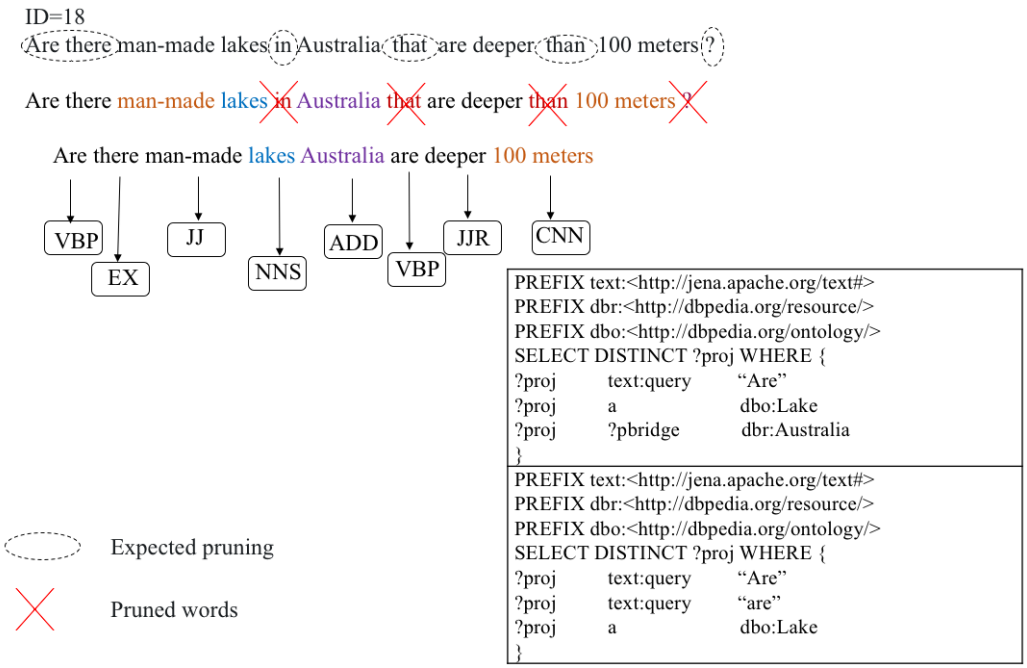


Figure 4.3. Example of linguistic pruning limitations – Question ID=18 extracted from QALD-7 Training Dataset

4.5 SEMANTIC ANNOTATION

Semantic annotation aims to find if nouns and verbs exist as classes or properties in the DBpedia ontology, and replaces them by their URIs when applicable. If not, these words are searched as expressions in DBpedia abstracts. However, finding the correct properties and classes is a challenge because of three main issues: duplicates of candidate annotations, lack of consistency of candidate annotations and quantity of candidate annotations.

Duplication. In some cases, a word is annotated either with the same URI several times or with several candidate classes or properties. This leads to the generation of several queries and hence takes a longer time to process and get the final answer. Figure 4.4 shows an example where *county* is mapped to four properties, and where *dbo:county* is duplicated.

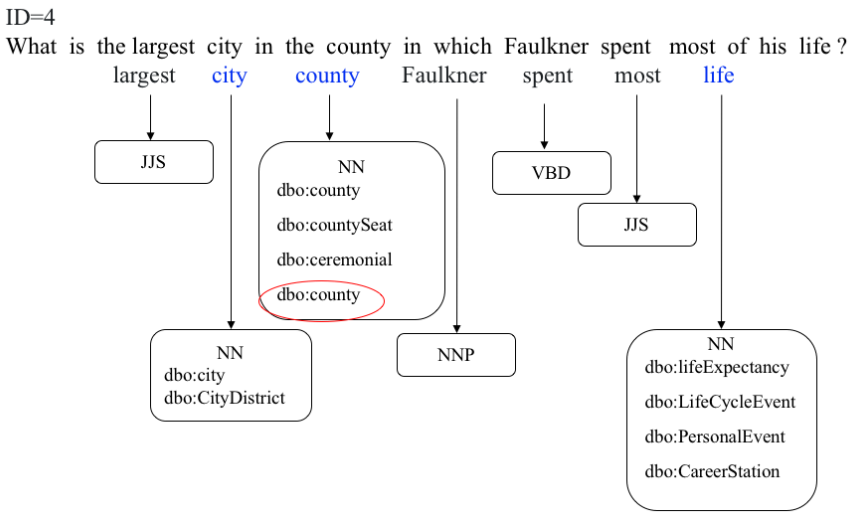


Figure 4.4. Example of semantic annotation duplication limitation – Question ID=4 extracted from QALD-7 Training Dataset

Consistency. The process of semantic annotation maps nouns and verbs to their URIs (classes, properties) in the ontology. However, some of the relevant classes or properties are ignored because of the ranking technique, which selects the most frequent property/class. One issue is that several irrelevant classes and properties can be selected as these elements are preserved or discarded based on their frequency of use in DBpedia and not based on the meaning of the question. This affects the final answer and can prevent from finding the correct answer. Figure 4.5 shows a case where the relevant property *dbo:birthDate* is ignored and *dbo:birthPlace* is kept instead.

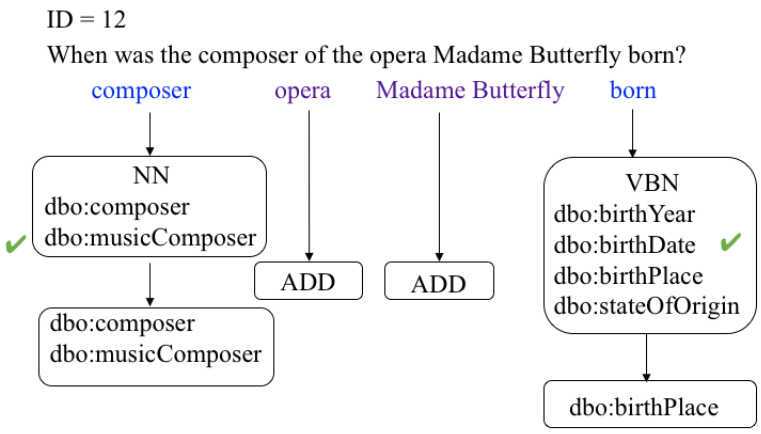


Figure 4.5. Example of semantic annotation consistency limitation – Question ID=12 extracted from QALD-7 Training Dataset

Quantity. In some cases, a word is annotated with several candidate classes and/or properties, which also increases the time required for processing queries. Figure 4.6 shows an example where "*members"* is mapped to 13 different classes.

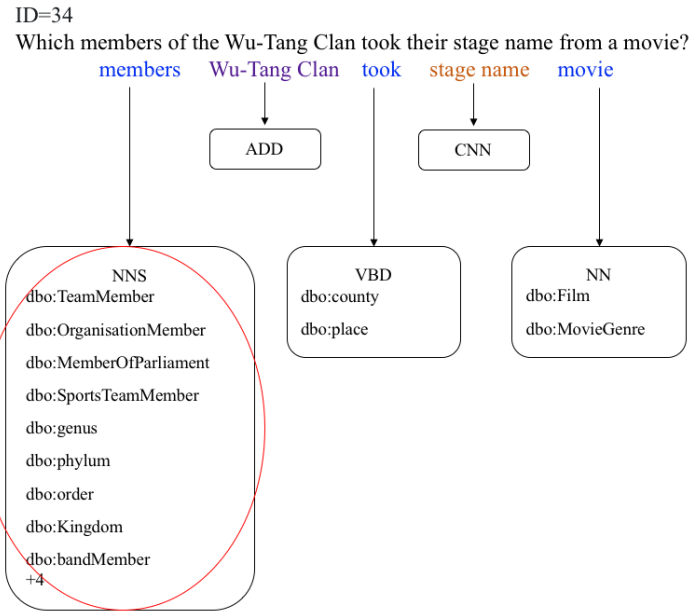


Figure 4.6. Example of Semantic Annotation Quantity limitation – Question ID=34 extracted from QALD-7 Training Dataset

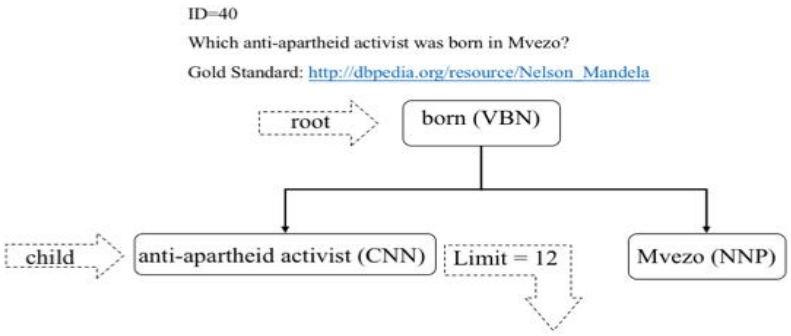
4.6 SPARQL Query Generation

A number of SPARQL queries are generated by combining the query fragments related to the question nodes (named entities, combined phrases, classes, properties and other words).

4.6.1 Cardinality Limit Identification

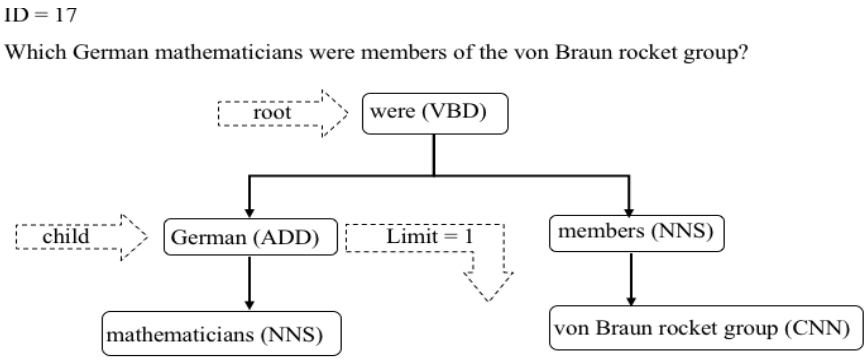
Queries are assigned a limit to restrict the number of returned answers. This cardinality number is identified based on the POS tag of the root or the POS tag of the left child of the root for a given question. In some cases, the assigned limit is inappropriate. For instance, a question has two correct answers, but HAWK assigns a query limit of 1, leading to missing correct answers. Moreover, the returned results of a SPARQL query are not ordered, and the correct answers might not be the first answers or among the first 12 answers in the answer list.

Figure 4.7 and Figure 4.8 show examples where the given cardinality numbers are incorrect and clarify the impact of the limit on the same SPARQL query.



LIMIT 12	LIMIT 1
PREFIX text:<http://jena.apache.org/text#> PREFIX dbr:<http://dbpedia.org/resource/> SELECT DISTINCT ?proj WHERE { ?proj text:query "anti-apartheid activist" ?proj text:query "University" ?proj ?pbridge dbr:South_Africa } LIMIT 12	PREFIX text:<http://jena.apache.org/text#> PREFIX dbr:<http://dbpedia.org/resource/> SELECT DISTINCT ?proj WHERE { ?proj text:query "anti-apartheid activist" ?proj text:query "University" ?proj ?pbridge dbr:South_Africa } LIMIT 1
http://dbpedia.org/resource/Tony_Karon http://dbpedia.org/resource/Nelson_Mandela http://dbpedia.org/resource/Amy_Biehl http://dbpedia.org/resource/Sheena_Duncan http://dbpedia.org/resource/Joe_Slovo,_Cape_Town http://dbpedia.org/resource/Bongani_Ndodana-Breen	http://dbpedia.org/resource/Tony_Karon

Figure 4.7. Example of cardinality limitation – Question ID=40 extracted from QALD-7 Training Dataset

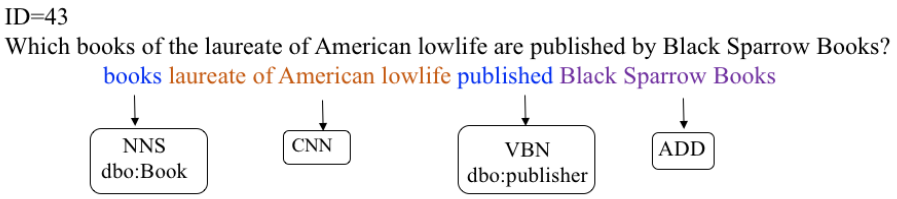


Final Answer	Gold Standard
http://dbpedia.org/resource/Oswald_Lange	http://dbpedia.org/resource/Walter_Haeussermann http://dbpedia.org/resource/Oswald_Lange

Figure 4.8. Example of cardinality limitation – Question ID=17 extracted from QALD-7 Training Dataset

4.6.2 *TEXT SEARCH*

Combined phrases are used in text search. However, these combined phrases might not be represented in Wikipedia pages/DBpedia abstracts as is, for example with other words between the combined words or in a different order. In these cases, *exact search* does not return any answer. *Fuzzy search*, which splits the expression into individual words, is then used, but it leads to another problem. HAWK looks for prepositions and determiners alone, which returns several answers that are not related to the question content. For example, many DBpedia abstracts might contain the preposition "of" without being related to the question content. In Figure 4.9, when the combined phrase "*laureate of American lowlife*" is searched using fuzzy search, each word of the expression is looked up individually. This means that the four words of the combined phrase can appear in any order in Wikipedia pages/DBpedia abstracts. For example, "*of*" is looked up anywhere in the text.



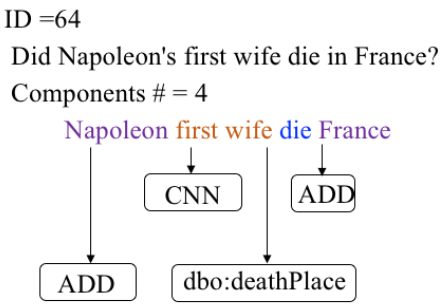
Exact text search	Fuzzy text search
PREFIX text:<http://jena.apache.org/text#> PREFIX dbr:<http://dbpedia.org/resource/ > PREFIX dbo:<http://dbpedia.org/ontology/> SELECT DISTINCT ?proj WHERE { ?proj text:query "laureate of American lowlife" ?proj rdf:type dbo:Book ?proj ?pbridge dbr:Black_Sparrow_Books } LIMIT 12	PREFIX text:<http://jena.apache.org/text#> PREFIX dbr:<http://dbpedia.org/resource/ > PREFIX dbo:<http://dbpedia.org/ontology/> SELECT DISTINCT ?proj WHERE { ?proj text:query "laureate~1 AND of ~1 AND American ~1 AND lowlife~1" ?proj rdf:type dbo:Book ?proj ?pbridge dbr:Black_Sparrow_Books } LIMIT 12
PREFIX text:<http://jena.apache.org/text#> PREFIX dbr:<http://dbpedia.org/resource/ > PREFIX dbo:<http://dbpedia.org/ontology/> SELECT DISTINCT ?proj WHERE { ?proj text:query "laureate of American lowlife" ?proj ?pbridge dbr:Black_Sparrow_Books } LIMIT 12	PREFIX text:<http://jena.apache.org/text#> PREFIX dbr:<http://dbpedia.org/resource/ > PREFIX dbo:<http://dbpedia.org/ontology/> SELECT DISTINCT ?proj WHERE { ?proj text:query "laureate~1 AND of ~1 AND American ~1 AND lowlife~1" ?proj ?pbridge dbr:Black_Sparrow_Books } LIMIT 12

Figure 4.9. Example of text search limitation – Question ID=43 extracted from QALD-7 Training Dataset

4.7 SEMANTIC PRUNING

Some SPARQL queries are pruned to eliminate queries that could negatively affect the score of the final answer. As previously explained, the queries are built by combining query fragments of each node (word, named entity, combined phrase and predicate) of the question. Some queries do not contain query fragments for all the required components. This means that the query does not include or cover all the required information and hence the answer will

necessarily be wrong. The question "*Did Napoleon's first wife die in France?*" consists of 4 nodes/components: "*Napoleon*", "*first wife*", "*die*" and "*France*". In Figure 4.10, it is clear that the query on the left does not return an accurate answer ("false") but might return the wife of Napoleon, which is not the aim of question.




```
PREFIX text:<http://jena.apache.org/text#>
PREFIX dbr:<http://dbpedia.org/resource/>
PREFIX dbo:<http://dbpedia.org/ontology/>
SELECT DISTINCT ?proj WHERE {
?proj     text:query     "first wife "
?proj     ?pbridge       dbr:Napoleon
?proj     ?pbridge       dbr:France
}

Query fragments # = 3
Components # = 4
=> Pruned
```

```
PREFIX text:<http://jena.apache.org/text#>
PREFIX dbr:<http://dbpedia.org/resource/>
PREFIX dbo:<http://dbpedia.org/ontology/>
SELECT DISTINCT ?proj WHERE {
?proj     text:query     "first wife "
?proj     a              dbo:deathPlace
?proj     ?pbridge       dbr:Napoleon
?proj     ?pbridge       dbr:France
}
Query fragments # = 4
Components # = 4
=> Not pruned
```

Figure 4.10. Example of semantic pruning limitation – Question ID=64 extracted from QALD-7 Training Dataset

4.8 Answer Verification

HAWK does not verify the adequacy of the returned answer with the answer type that the question is seeking. An example appears in the question "*Who are the architects of the tallest building in Japan?*" where the clue is "*architects*" (see Figure 4.11). However, HAWK returns an answer set composed of the following resources: *dbr:César_Pelli* (an architect), *dbr:Nikken_Sekkei* (an organization) and *dbr:Takenaka_Corporation* (a company).

ID=46
Who are the architects of the tallest building in Japan?

Final Answer	Gold Standard
http://dbpedia.org/resource/César_Pelli http://dbpedia.org/resource/Nikken_Sekkei http://dbpedia.org/resource/Takenaka_Corporation	http://dbpedia.org/resource/César_Pelli http://dbpedia.org/resource/Nikken_Sekkei

Expected answer type

Figure 4.11. Example of answer type limitation – Question ID=46 extracted from QALD-7 Training Dataset

4.9 RANKING

For each question, several queries are fired. These queries return a list of answer sets ranked by their frequencies. However, only one answer set has to be returned as the final correct answer. While some questions have the correct answer in their answer sets, their ranking is sometimes incorrect. Figure 4.12 shows an example of this case where the correct answer *dbr:Paris* has a frequency lower than other answer sets, and hence it is not returned as the final answer.

ID=41
The home of the Mona Lisa is a landmark of which city?

Answer sets	Frequency	
http://dbpedia.org/resource/Liverpool	30	Final answer
http://dbpedia.org/resource/Scotland	4	
http://dbpedia.org/resource/Paris	2	Correct answer

Figure 4.12. Example of overlap-based ranking limitation – Question ID=41 extracted from QALD-7 Training Dataset

4.10 CONCLUSION

We examined HAWK's pipeline in detail and identified several limitations. As described above, some issues exist in almost every step, except the first step of POS tagging. The main impact of these issues is on the correctness of the final answer and on the number of generated queries. Hence, the accuracy of the answers and the speed of answering a question are

negatively affected. In the next chapter, we propose some solutions and modifications to the above-mentioned issues to improve HAWK's performance.

CHAPTER 5. HAWK_R

5.1 CHAPTER OVERVIEW

In this chapter, we propose solutions to overcome HAWK's aforementioned limitations. The modifications applied to each module are illustrated by examples to show how these changes affect the performance. We show the impact of each modified module on the overall performance by changing only one module at a time. These modifications are tested on several datasets, i.e. QALD-4 to QALD-6 training and test datasets in addition to QALD-7 training dataset. Note that to lighten the chapter's content, the detailed results of each modification are shown in the annex. The steps that remain unchanged compared to the original system HAWK are omitted. Finally, we draw conclusions from these individual experiments and we combine the most successful modifications in a system called HAWK_R.

5.2 QUESTION TYPE CLASSIFICATION

HAWK classifies questions into Boolean and resource questions, thus it cannot answer all the potential questions such as date and numeric questions. To overcome this issue in HAWK_R, we implemented a more fine-grained classification of the question types, which enables the verification of the answer. This question type classification is based on lexical patterns (See Table 5.1) that assign the question to a type that is either a class (dbo:Person) or a typed literal (xsd:gYear). For instance, the question "*Where did the first man in space die?*", is mapped to the class *dbo:Place*. The question "*In which year did the Hungarian-American actor called "The King of Horror" make his first film?*" is classified as a Year question and is mapped to *xsd:gYear*.

Table 5.1. Lexical patterns for question type classification

Pattern	**Question Type**	**Class / Literal**
Auxiliary verbs	Boolean	true / false
When What / Which date Clue: xsd:date	Date	xsd:date

What/Which year Clue: xsd:date	Year	xsd:gYear
How many/much/old *with* Clue: xsd:positiveIntege	Numeric	xsd: positiveInteger
How many/much/old *with* Clue: owl:ObjectProperty	Count	
How many/much/old *with* Clue: Empty annotation	Text-extraction	
Who, Whose	Resource (Person)	dbo:Person
Where What / Which place Clue: dbo:Place	Resource (Place)	dbo:Place
What, Which, Give, List *with* Clue: owl:ObjectProperty	Resource (Thing)	owl:Thing
What, Which, Give, List *with* Clue: owl:Class		
No question word	String	String

Questions are classified based on the question word and the context of the question (clue). The clue represents a keyword in the question that determines the answer type, which is used for the classification. It is the first verb or noun that occurs after the question word. For example, the clue of the question "*A landmark of which city is the home of the Mona Lisa?*" is "*city*". We rely on DBpedia and WordNet[32] to find the clue type. We check the annotation of the clue obtained during the semantic annotation stage and find its DBpedia type. For clues that are not mapped to any class or property from DBpedia, we rely on WordNet to find the category of the clue. Then, we find the class or predicate in DBpedia that can be mapped to that clue category based on a simple match of the rdfs:label. For instance, in the question "*Which anti-apartheid activist was born in Mvezo?*", the clue "*activist*" is not found in DBpedia. Based on WordNet, we find that the category of "*activist*" is "Person", which is mapped to *dbo:Person.*

We use the DBpedia types of the word and clue (annotation) to classify the questions. For example, a question that contains the question word "*When*" or includes "*What/Which date*",

[32] https://wordnet.princeton.edu/

or whose clue type is a datatype property with a range of *xsd:date* is classified as a *Date* question. For questions that are classified as *Count* questions, HAWK_R counts the answers and returns a number as a result of the query. For questions that are classified as *Text-extraction*, HAWK_R analyzes the text to extract the answer that matches the clue. For instance, for the question "*How old was Steve Job's sister when she first met him?*", HAWK_R analyzes the abstract of *dbr:Mona_Simpson* [*Simpson was born after her parents had married and she did not meet Jobs until she was 25 years old*] to extract the answer which is (25). Figure 5.1 shows an example of a case before and after modification.

ID = 99
How old was Steve Jobs' sister when she first met him?
HAWK question classification: Resource
HAWK_R question classification: Text-extraction
Clue: old
Gold standard answer: 25

HAWK answer	HAWK_R answer
http://dbpedia.org/resource/American_Broadcasting_Company http://dbpedia.org/resource/The_Bay_(radio_station) http://dbpedia.org/resource/ABC_(Australian_TV_channel) http://dbpedia.org/resource/Rede_Bandeirantes http://dbpedia.org/resource/Discovery_Family http://dbpedia.org/resource/KING-TV http://dbpedia.org/resource/Capital_(radio_network) http://dbpedia.org/resource/Disney_Channel http://dbpedia.org/resource/WBEZ http://dbpedia.org/resource/CBS_Europa http://dbpedia.org/resource/Fusion_(TV_channel) http://dbpedia.org/resource/Watt	25

Figure 5.1. Example of improved question type classification – Question ID=99 extracted from QALD-7 Training Dataset

5.3 PHRASE COMBINATION

Phrase combination aims to combine tokens that are not recognized as named entities in the named entity recognition phase. As described in HAWK's limitations, combination rules have three issues that could lead to incorrect answers. To overcome the problems, adjectives are combined with noun phrases, whereas the combination of quantity words and prepositions with noun phrases is avoided. The altered rules are shown in Figure 5.2.

StopWords= {many, much}
1. Token [CD|JJ(.)*|NN(.)*|RB(.)*] ∉ StopWords
2. Token [IN] NextToken [(W)?DT|NNP(S)?")]
3. Token [NNP(S)?]
4. Token [VB(.)*|.|WDT] || Token [IN] NextToken [NN(.)*]
5. Token [NN(.)* | RB | CD | CC | JJ(.)* | DT | PRP | HYPH | VBN]

Figure 5.2. Improved heuristic rules of phrase combination

Figure 5.3 presents some examples of questions after modifying the combination rules. Adjectives and adverbs are combined with noun phrases such as "*tallest building*" and "*famous equation*". Moreover, quantity words are not anymore combined with phrases such as "*many awards*" in the second question. Finally, we add the rule that combined phrases cannot end with prepositions as highlighted in the third question where "*capital*" is not combined with "*of*", because it restricts the possibility of mapping a word to a semantic annotation in a later stage.

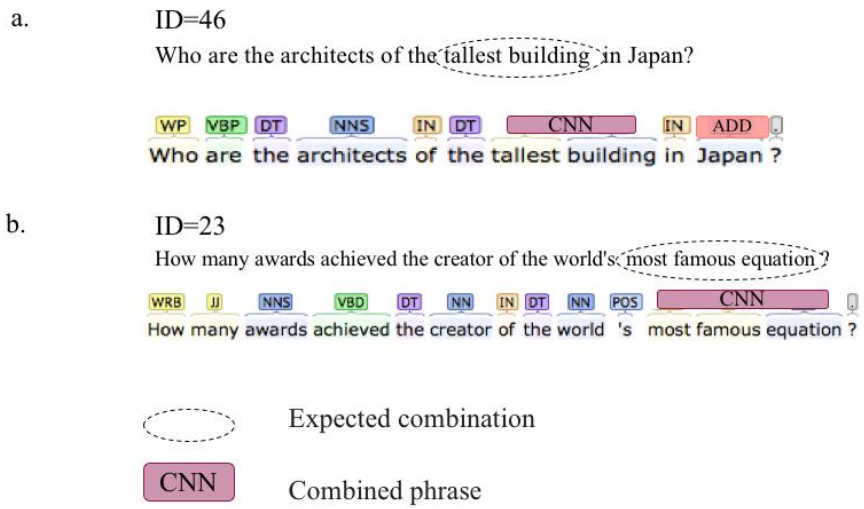


Figure 5.3. Example of improved phrase combination – Question ID=46, 23 extracted from QALD-7 Training Dataset

5.4 LINGUISTIC PRUNING

The purpose of linguistic pruning is to remove words that could produce noise and affect the final answer, or that could generate incorrect answers. Simply, we prune determiners, prepositions, auxiliary verbs and question words (after the question classification stage). Figure 5.4 demonstrates the improvement of this process by presenting examples where

question words such as "*Who*", auxiliary verbs such as "*are*", prepositions such as "*than, in, by, around*" and determiners such as "*the*" are pruned.

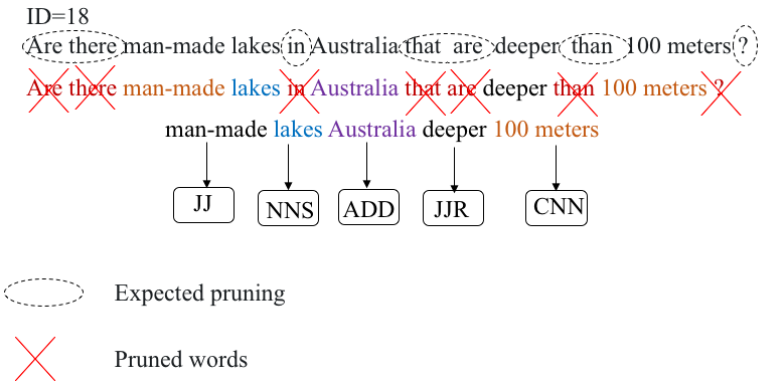


Figure 5.4. Example of improved linguistic pruning – Question ID= 0 extracted from QALD-7 Training Dataset

5.5 SEMANTIC ANNOTATION

Semantic annotation aims to map nouns and verbs to classes and properties in the DBpedia ontology. As described in HAWK's limitations, this step has three issues that could lead to incorrect answers. For the problem of duplicates, we simply eliminate duplicate annotations. For consistency issues, semantic annotations are verified based on their coherence and relatedness to the context of the question. In short, candidate annotations are checked (as explained next) to be considered as *verified annotations*. For the ranking problem, annotations are ranked based on their frequency of use in DBpedia, and the most popular one is considered a *final annotation*.

Figure 5.5 demonstrates the process of verifying candidate annotations and ranking verified annotations in order to find the final annotations.

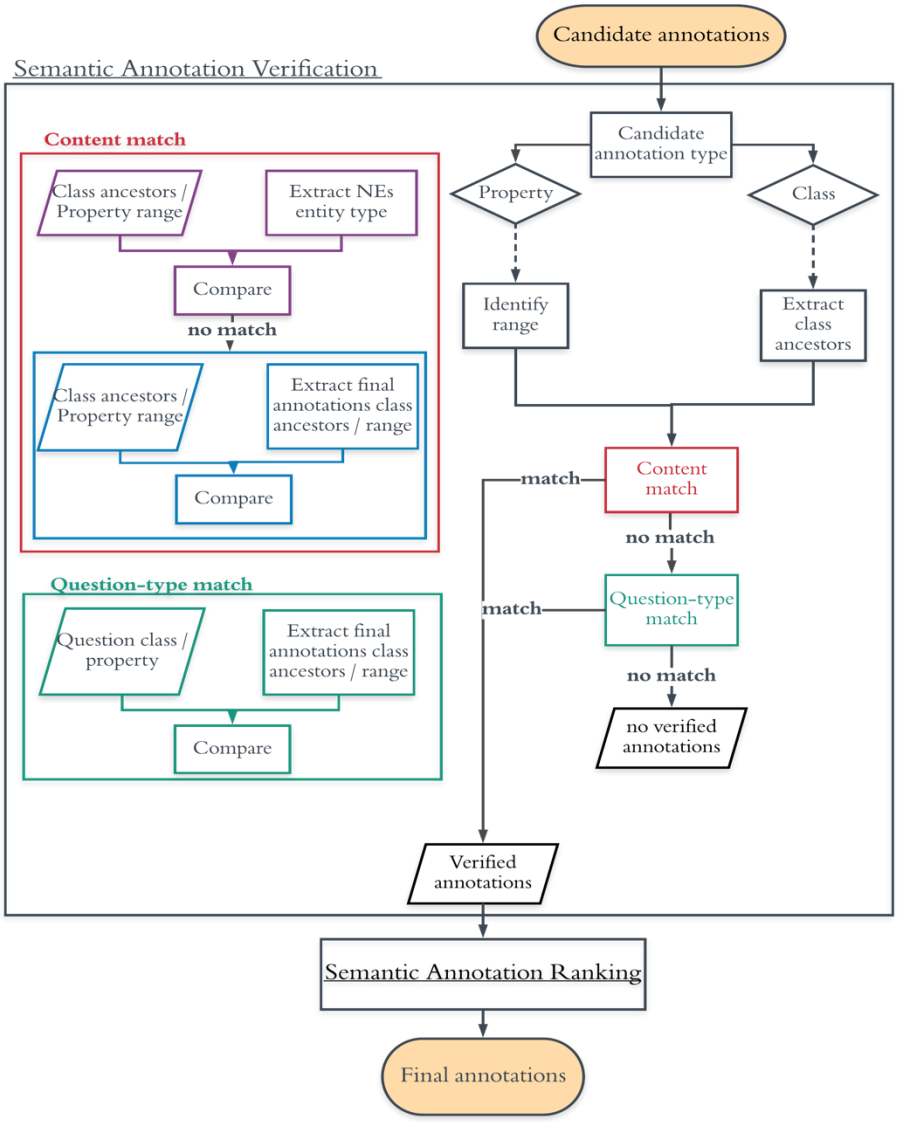


Figure 5.5. Flow chart of semantic annotation filtering and ranking

5.5.1 SEMANTIC ANNOTATION VERIFICATION

The objective in this step is to obtain coherent types among all the annotations assigned to the question components. Candidate annotations are considered coherent if they are homogeneous (in terms of types) with other question components such as named entities. For every annotation, we extract its types (i.e. classes) using the *rdf:type* property values along with the hierarchy of the classes. Then, we compare these types with the types of the question's named entities and question type class to check the consistency of the candidate annotation.

5.5.1.1 Candidate Annotation Type

Class Candidate Annotation: For each expression annotated as a class, if the only ancestor of the class is owl:Thing, then we use the class itself. For instance, the ancestor of *dbo:Place* is *owl: Thing*. Therefore, we keep *dbo:Place* only.

Otherwise, we extract the full hierarchy of the class (the ancestors) until we reach owl:Thing. For instance, the SPARQL query `SELECT ?subClass { dbo:Settlement a owl:Class; rdfs:subClassOf+ ?subClass. }` returns the ancestors set [*dbo:PopulatedPlace, dbo:Place, owl:Thing*] as ancestor classes for *dbo:Settlement*.

Property Candidate Annotation: For each expression annotated as a property, we identify its range. For instance, the *rdf:type* of *dbo:birthPlace* is a property and its range is the class *dbo:Place* whereas the range of *dbo:birthDate* is *xsd:date*.

5.5.1.2 Consistency Check

We use the obtained class ancestors and property range to check the consistency at two different levels: *Content match* and *Question-type match*. We first check the coherence of a candidate annotation with the identified named entities and with already identified final annotations in the question. Finally, we check the consistency against the *Question type* if we do not find any match in the previous phase. If at any phase we find a match, we consider the annotation as a verified annotation.

***Content Match*:** In this phase, we aim at obtaining coherent types among all the annotations assigned to the question components. Types are considered coherent if they are the same or if they belong to the same class hierarchy. We check all the annotations against all the identified named entities. This is done by comparing the class ancestors / property range of each annotation with all the entity types of each named entity. For instance, in the question "*Did Napoleon's first wife die in France?"* in Figure 5.6, there are two named entities *Napoleon* and *France* and one predicate *die.* The named entities have the following types:

- *dbr:Napoleon* : [*dbo:Person, dbo:Agent, dbo:Royalty*].
- *dbr:France* : [*owl:Thing, dbo:Place, dbo:Country*].

When we try to determine the annotation of the verb "*die*", we end up with several candidate annotations [*dbo:deathPlace, dbo:deathDate, dbo:deathYear*]. Given that we find a match between the range of the property *dbo:deathPlace* (*dbo:Place*) and the types of *dbr:France*, the property *dbo:deathPlace* becomes a verified annotation. The other properties *dbo:deathDate* and *dbo:deathYear* are ignored as we do not find any match with the types of *dbr:Napoleon* and *dbr:France*.

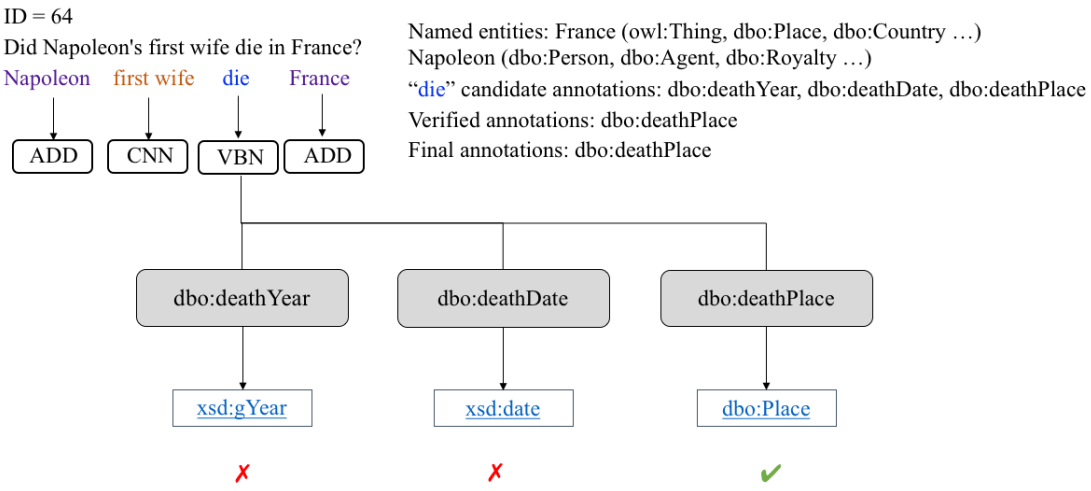


Figure 5.6. Example of improved semantic Annotation – Question ID=64 extracted from QALD-7 Training Dataset

If we do not find any match with named entities, we compare the candidate annotations with the already identified final annotations in the question in a similar matching process. For instance, in the question "*Which country did the first known photographer of snowflakes come from?*" in Figure 5.7, "*country*" is linked to the class *dbo:Country* and we want to verify the consistency of the candidate annotations *dbo:birthPlace* and *dbo:releaseDate* of the verb "*come*". Therefore, we compare the class hierarchy of *dbo:Country* which is [*dbo:PopulatedPlace, dbo:Place, owl:Thing*] with the range of *dbo:birthPlace* which is [*dbo:Place*] and the range of *dbo:releaseDate* which is [*xsd:date*]. Since we find a match between *dbo:Country* and *dbo:birthPlace*, *dbo:birthPlace* becomes a verified annotation.

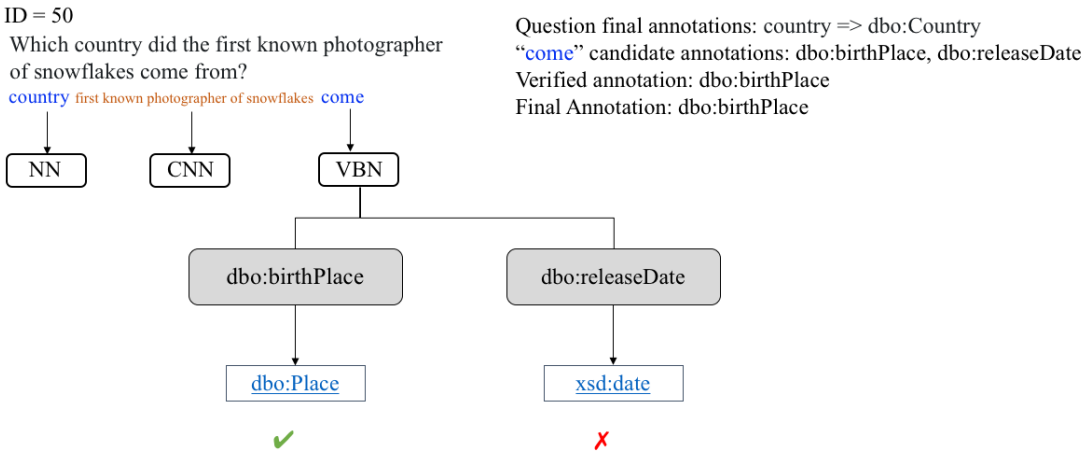


Figure 5.7. Example of improved semantic Annotation – Question ID=50 extracted from QALD-7 Training Dataset

***Question-type Match*:** The class/property of the question type (section 5.4) is used to filter the candidate annotations in a matching process similar to the content match. The class/property of the question type is compared to the class hierarchy/property's range of the candidate annotation to find a match. In the question "*Where did the first man in space die?*" in Figure 5.8, we find a match between the type of the question and the range of the property *dbo:deathPlace*. Therefore, *dbo:deathPlace* becomes a verified annotation for "*die*".

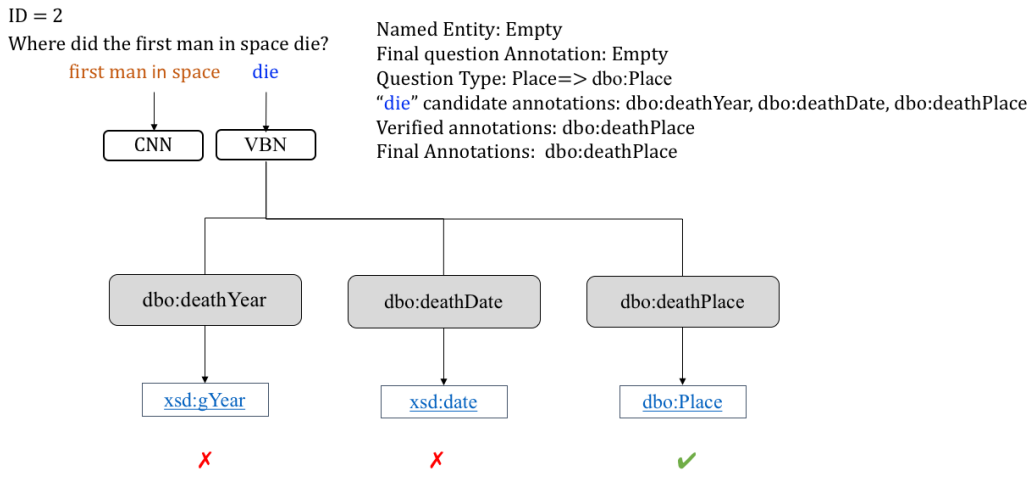


Figure 5.8. Example of improved semantic Annotation – Question ID=2 extracted from QALD-7 Training Dataset

5.5.2 SEMANTIC ANNOTATION RANKING

After filtering the annotations, the verified annotations are ranked based on their frequency of use in DBpedia, and the most frequent one is chosen. This means that we favor classes or properties that are more "popular". For instance, in the question "*Where was the "Father of Singapore" born?*", the verb "*born*" has several candidate annotations [*dbo:birthDate, dbo:birthYear, dbo:birthplace, dbo:stateOfOrigin*] among which only two are verified annotations [*dbo:birthplace, dbo:stateOfOrigin*]. Finally, *dbo:birthPlace* is the final annotation because it has a frequency higher than *dbo:stateOfOrigin*.

Properties are more commonly used as predicates in DBpedia triples. For instance, the answer to the question "*Which horses did The Long Fellow ride?*" can be obtained using the property *dbo:raceHorse.* However, the final annotation of "*horses*" is the class *dbo:RaceHorse.* For that, we use a SPARQL query to find the range of the class *dbo:RaceHorse.* At the end, both the class *dbo:RaceHorse* and the property *dbo:raceHorse* are final annotations of "*horses*".

5.6 SPARQL QUERY GENERATION

In this stage, HAWK_R generates SPARQL queries and has to first distinguish between questions that require a single answer and questions that require several answers.

5.6.1 CARDINALITY LIMIT IDENTIFICATION

The assigned cardinality number is used to limit the number of returned answers per query based on the expected number of answers.

Identifying Optimal Limit: HAWK arbitrarily sets the limit number to 12 for questions that have multiple answers and to 1 otherwise. However, we noticed that often the correct answer was not necessarily among the first 12 answers. To overcome this issue, we tested different limit numbers on the QALD-7 training dataset to determine the optimal limit number that would not eliminate any correct answer. We compared several limit values: 12, 22, 100 and No Limit. We found that when the limit is set to 100, the F-measure increases compared to other suggested limit values. The cardinality that is identified for each question is used later in Answer Verification to return one or more answers for each answer set. For instance, in the question "*Which anti-apartheid activist was born in Mvezo?*", we first choose a limit 100 although the cardinality is 1 based on the analysis of the question. Then, in a later stage, we

use this cardinality to keep one or several resources in the answer set. The performance of the system when using the different suggested limit values is shown in the annex.

Assigning a Limit: HAWK policy to assign this limit relies on the root of the dependency parse tree. In our case, we propose the use of lexico-syntactic patterns to identify the proper limit based on the analysis of the question. The heuristic rules for identifying the cardinality number based on the clue, which is the first occurrence of a noun or verb after the question word, are shown in Table 5.2.

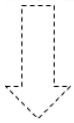
Table 5.2. Lexical patterns of cardinality identification

Pattern	**Example**	**Cardinality**
People with question word Whose	*Of the people that died of radiation in Los Alamos, whose death was an accident?*	100
Clue is a plural noun	*Which German mathematicians were members of the von Braun rocket group? (see Figure 5.9)*	100
Clue is a singular noun that is followed by a plural noun	*list all the horror movies that won awards in 2010.*	100
Clue is a singular noun	*For which movie did the daughter of Francis Ford Coppola receive an Oscar?*	1
Clue is a singular noun that is followed by a singular noun	*Which street basketball player was diagnosed with Sarcoidosis?*	1
Clue is a verb	*Who played the drums in the band that wrote Complete Control?*	1

The question in figure 5.9 shows a case where the question is seeking more than one mathematician. HAWK assigns 1 as a cardinality number based on the child (*German->ADD*) of the root (*were -> VBD*). However, the limit in HAWK_R is 100 based on the POS tag of the clue (*mathematicians -> NNS*).

ID = 17

Which German mathematicians were members of the von Braun rocket group?



Limit = 12

Final Answer	Gold Standard
http://dbpedia.org/resource/Walter_Haeussermann http://dbpedia.org/resource/Oswald_Lange	http://dbpedia.org/resource/Walter_Haeussermann http://dbpedia.org/resource/Oswald_Lange

Figure 5.9. Example of improved cardinality – Question ID= 17 extracted from QALD-7 Training Dataset

5.6.2 TEXT SEARCH

Combined phrases and words that are not found in the DBpedia ontology in the form of properties or classes are looked up in Wikipedia pages/DBpedia abstracts. As previously explained, text search is performed using two different methods: Exact text search and fuzzy text search.

For both exact and fuzzy text search, we not only look for the textual expression as is, but we also apply lemmatization to account for various forms of the expression (e.g. "*buildings*" / "*building*"). Both elements are looked up in Wikipedia pages/DBpedia abstracts. Moreover, we ignore prepositions and determiners in the fuzzy search. Figure 5.10 shows an example of a text search query before and after the modification. The combined phrase "*laureate of American lowlife*" is split into four words "*laureate*", "*of*", "*American*" and "*lowlife*". The preposition "*of*" is simply ignored.

ID=43

Which books of the laureate of American lowlife are published by Black Sparrow Books?

books laureate of American lowlife published Black Sparrow Books

Fuzzy text search (Original)	Fuzzy text search (Modified)
PREFIX text:<http://jena.apache.org/text#> PREFIX dbr:<http://dbpedia.org/resource/ > PREFIX dbo:<http://dbpedia.org/ontology/> SELECT DISTINCT ?proj WHERE { ?proj text:query "laureate~1 AND of ~1 AND American ~1 AND lowlife~1" ?proj rdf:type dbo:Book ?proj ?pbridge dbr:Black_Sparrow_Books } LIMIT 12	PREFIX text:<http://jena.apache.org/text#> PREFIX dbr:<http://dbpedia.org/resource/ > PREFIX dbo:<http://dbpedia.org/ontology/> SELECT DISTINCT ?proj WHERE { ?proj text:query "laureate~1 AND American ~1 AND lowlife~1" ?proj rdf:type dbo:Book ?proj ?pbridge dbr:Black_Sparrow_Books } LIMIT 12
PREFIX text:<http://jena.apache.org/text#> PREFIX dbr:<http://dbpedia.org/resource/ > PREFIX dbo:<http://dbpedia.org/ontology/> SELECT DISTINCT ?proj WHERE { ?proj text:query "laureate~1 AND of ~1 AND American ~1 AND lowlife~1" ?proj ?pbridge dbr:Black_Sparrow_Books } LIMIT 12	PREFIX text:<http://jena.apache.org/text#> PREFIX dbr:<http://dbpedia.org/resource/ > PREFIX dbo:<http://dbpedia.org/ontology/> SELECT DISTINCT ?proj WHERE { ?proj text:query "laureate~1 AND American ~1 AND lowlife~1" ?proj ?pbridge dbr:Black_Sparrow_Books } LIMIT 12

Figure 5.10. Example of improved fuzzy search – Question ID=43 extracted from QALD-7 Training Dataset

5.7 SEMANTIC PRUNING

Semantic pruning is used to reduce the number of useless SPARQL queries and to increase the possibilities of having correct answers. Therefore, to ensure that each executed query covers all the important required components of the question, we propose a new pruning technique based on the idea of pruning any query whose number of query fragments is less than the number of the question components. Figure 5.11 clarifies this concept using an example. The question consists of four components: "*Napoleon"*, "*first wife*", "*die*" and "*France*". Therefore, each SPARQL query must have four query fragments, one for each component, to make sure that the query covers all the aspects of the question. Based on this idea, the query on the left is pruned because it does not cover all the question content while the query on the right is not pruned.

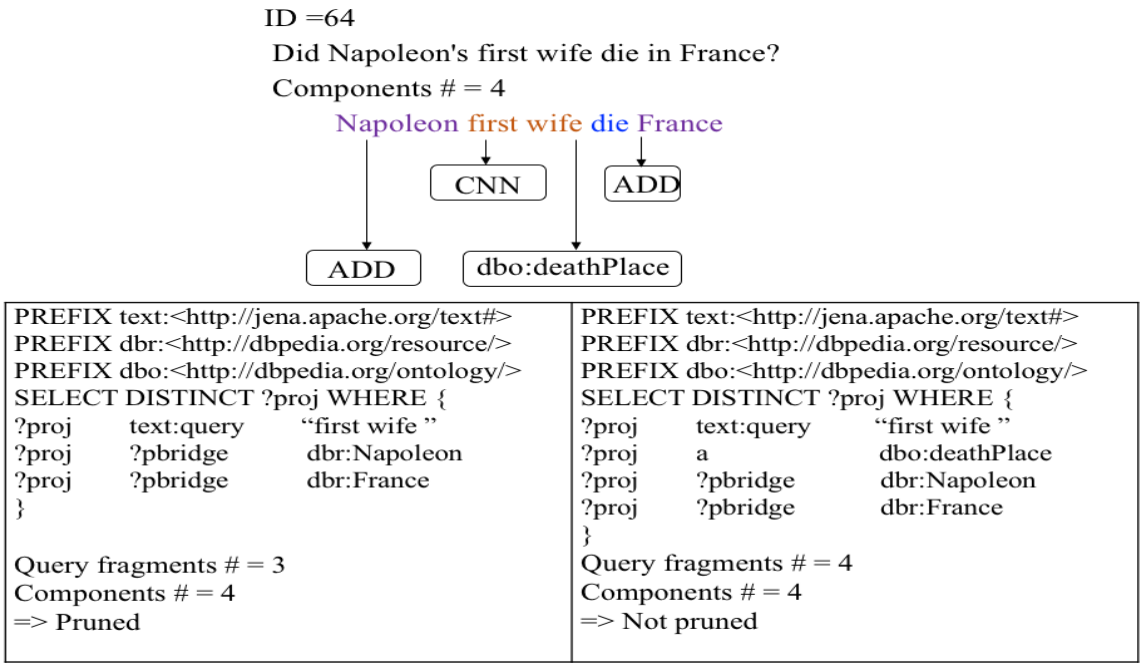


Figure 5.11. Example of the new pruning technique – Question ID=64 extracted from QALD-7 Training Dataset

5.8 Answer Verification

For each question, a collection of answer sets is returned. These answer sets include wrong and right answers. One way to ensure that the final answer is correct is to guarantee that the final answer has a correct type.

For that, we implemented a new functionality in HAWK_R that verifies the results' coherence and relatedness with the clue type and the question type (identified in Question Type Classification). The overall process is the following: we first check the coherence of the results with the clue type (Clue-type match). If we do not find any match, we compare the results with the question type (Question-type match).

5.8.1 Cleaning Answer Set and Identifying Answer Type

For each answer set, we remove the answers that match a named entity in the question. For instance, one answer set for the question "*A landmark of which city is the home of the Mona Lisa?*" is [*dbr:Mona_Lisa, dbr:Paris*]. After pruning, the answer set becomes [*dbr:Paris*].

Then, we find the class hierarchy of all the resources in the answer set. For instance, the class hierarchy of *dbr:Paris* is [*dbo:PopulatedPlace, dbo:Place*]. We use this class hierarchy to check the coherence of the answer with the clue type and question type.

5.8.2 VERIFYING ANSWERS

5.8.2.1 Clue-type Match

In HAWK_R, query results are verified based on their consistency and relatedness to the clue type. We extract the class hierarchy of the clue type identified in question classification. After obtaining the clue hierarchy (CH), we use it to check the coherence of the query results. For each answer of the answer set, we compare the class hierarchy of the answer with the class hierarchy of the clue. For example, for the question "*A landmark of which city is the home of the Mona Lisa?*" the CH is [*dbo:City, dbo:Settlement, dbo:PopulatedPlace, dbo:Place*]. This process allows us to verify if there is a type match with the types of the candidate answers *dbr:Athens, dbr:Austin_Peay_State_University* and *dbr:Paris*. The resources that are consistent with the CL are *dbr:Athens and dbr:Paris*, which become verified answers, see Figure 5.12.

5.8.2.2 Question-type Match

If none of the answers in the answer set has a match with the CH, then we compare the class hierarchy of the answers with the class/property of the question (see Table 5.1). The objective is to identify whether any of the answers has the same type as the question. For example, one of the answers to the question "*Who plays Phileas Fogg in the adaptation of Around the World in 80 Days directed by Buzz Kulik?*" is *dbr:Pierce_Brosnan* and its class hierarchy is [*dbo:Person, dbo:Thing*]. When we compare this class hierarchy with the CL [*dbo:starring, dbo:Actor*] of "*plays*" we do not find any match. However, we find a match when we compare the class hierarchy of the answer with the question's class *dbo:Person*. Thus, the answer becomes a verified answer and we move to the next candidate answer.

For questions that are classified as *Count* questions, HAWK_R counts the verified answers and returns a number as a result as in the question "*How many awards achieved the creator of the world's most famous equation?*". For questions that are classified as *Text-extraction*, HAWK_R analyzes the text to extract the answer that matches the clue. For instance, one

answer for the question "*How old was Steve Jobs' sister when she first met him?*" is in the abstract of *dbr:Mona_Simpson*. HAWK_R analyzes the abstract sentence by sentence to extract the closest number to the word "old". In our case, we find the answer in the sentence: [*Simpson was born after her parents had married and she did not meet Jobs until she was **25** years **old**.*].

5.8.3 RANKING ANSWERS

After verifying all the answers of the answer set, we check the question cardinality (identified in an earlier stage) to validate the number of answers in that specific answer set. If the cardinality number of the question is one, then each answer set must have one answer only. To do that, we rank verified answers based on their frequency of use (as subjects) in DBpedia. The most frequent verified answer in the answer set becomes the final answer of the specified answer set. For instance, the answer set [*dbr:Athens, dbr:Paris*] of the example in Figure 5.12 consists of two verified answers. Using the SPARQL query `SELECT DISTINCT COUNT(?o) {dbr:Paris ?p ?o.}`, we find that the frequency of *dbr:Paris* is higher than the frequency of *dbr:Athens*. Thus, *dbr:Paris* is the final answer.

ID=10
A landmark of which city is the home of the Mona Lisa?
Gold standard answer: dbr:Paris
Clue: city => dbo:city
Clue class hirearchy: dbo:City, dbo:Settlement, dbo:PopulatedPlace, dbo:Place
Answer set: [dbr:Athens, dbr:Austin_Peay_State_University, dbr:Paris]

Before Answer Verification and Ranking			**After Answer Verification and Ranking**	
Candidate answers	Frequency	rdf:type	Verified answers	Frequency
dbr:Athens	3	dbo:Place dbo:Settlement dbo:Region owl:Thing	dbr:Athens	3
dbr: Austin_Peay_State_University	6	dbo:University dbo:Organisation dbo:Agent owl:Thing	Answer not compliant with our coherence check.	
dbr:Paris	5	dbo:Place dbo:Settlement dbo:Region owl:Thing	dbr:Paris	5

Final answer

Final answer

Figure 5.12. Example of Answer verification – Question ID=13 extracted from QALD-7 Training Dataset

5.9 TRIPLE-BASED RANKING

As previously explained, one major limitation in HAWK is the fact that the ranking of final results relies solely on frequency. In HAWK_R, we propose a new ranking technique (*Triple-based ranking)* that utilizes the triples of the query as features to score each query using a **query confidence score**. We assign empirical confidence scores to query triples as shown in Table 5.3 based on the triple content (resource, property, class, exact text search, fuzzy text search and variable). The values of these scores determine the importance of a particular type of triple.

Table 5.3. Query fragments scoring rules

Feature	**Example**	**Score**
Resource	?proj ?pbridge dbr:France	4
Property	?proj dbo:birthPlace ?const	4
Class	?proj a dbo:Place	4
Exact text search	?proj text:query “country member”	3
Fuzzy text search	?proj text:query “country~1 AND member~1”	2
Variable	?proj ?proot ?const	1

Triples that contain a DBpedia resource, class or property have higher scores than other triples as they have been determined through a complex semantic analysis of the question. Moreover, triples that include exact text search are given higher scores than triples of fuzzy text search. In fact, in fuzzy search, the words of the combined phrase might appear separately and do not have to appear in sequence. The overall confidence score of the query is the sum of the scores of the query triples.

We also compute a confidence score for each answer set called the **answer confidence score**. This score is the sum of scores of all queries that give the same answer set. The final answer is the answer set that has the highest answer confidence score. For instance, in Figure 5.13, each answer set is returned by two queries. Since the confidence score of the answer set

[*dbr:Barton_Fink*] is higher than the confidence score of [*dbr:Fargo_(film)*], HAWK_R returns [*dbr:Barton_Fink*] as the final answer set for the question "*Which movie by the Coen brothers stars John Turturro in the role of a New York City playwright?*".

ID=5
Which movie by the Coen brothers stars John Turturro in the role of a New York City playwright?

Answer set	Query	Query confidence score	Answer confidence score

[dbr:Barton_Fink]	PREFIX text:<http://jena.apache.org/text#>
	SELECT DISTINCT ?proj WHERE {
	?proj text:query "Coen brothers stars" => 4
	?proj text:query "John Turturro" => 4
	?proj ?proot ?const => 1
	}
		9	9 + 10 = 19 √

	PREFIX text:<http://jena.apache.org/text#>
	PREFIX dbo:<http://dbpedia.org/ontology/>
	SELECT DISTINCT ?proj WHERE {
	?const text:query "New York City playwrite" => 4
	?proj a dbo:Film => 5
	?proj ?proot ?const => 1
	}
		10	

[dbr:Fargo_(film)]	PREFIX text:<http://jena.apache.org/text#>
	SELECT DISTINCT ?proj WHERE {
	?proj text:query "Coen brothers stars" => 4
	?proj ?proot ?const => 1
	}
		5	5 + 9 = 14 X

	PREFIX text:<http://jena.apache.org/text#>
	SELECT DISTINCT ?proj WHERE {
	?proj text:query "Coen brothers stars" => 4
	?proj text:query "role" => 4
	?proj ?proot ?const => 1
	}
		9	

Figure 5.13. Example of query scoring and answer ranking – Question ID=5 extracted from QALD-7 Training Dataset

5.10 Evaluation of the Proposed Features

In order to overcome HAWK's limitations, we modified some existing features and implemented new features as well. Five features have been modified: Question Type Classification, Phrase combination, Linguistic Pruning, Text Search and Cardinality Limit Identification. Moreover, three new features have been proposed: Semantic Annotation and Verification, Answer Verification and Triple-based ranking. Table 5.4 and Table 5.5 recapitulate the performance of HAWK_R with the modification of each step over different datasets using HAWK's original ranking strategies: overlap-based ranking and optimal ranking.

Table 5.4. Comparison of HAWK_R features' performance using overlap-based ranking

QALD Challenge	Measures	HAWK	Question Type Classification	Phrase Combination	Linguistic Pruning	Semantic Annotation Verification	Text Search	Cardinality Limit Identification	Semantic Pruning	Answer Verification	Triple-based Ranking
QALD-4 Training Dataset	Recall	0.059	0.079↑	0.031↓	0.063↑	0.091↑	0.059↔	0.051↓	0.004↓	0.057↓	0.059↔
	Precision	0.090	0.110↑	0.067↓	0.130↑	0.147↑	0.090↔	0.087↓	0.040↓	0.095↑	0.090↔
	Macro F-measure	0.071	0.092↑	0.042↓	0.085↑	0.112↑	0.071↔	0.064↓	0.007↓	0.071↔	0.071↔
QALD-4 Test dataset	Recall	0.100	0.100↔	0.200↑	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔
	Precision	0.100	0.100↔	0.200↑	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔
	Macro F-measure	0.100	0.100↔	0.200↑	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔
QALD-5 Training Dataset	Recall	0.054	0.054↔	0.042↓	0.054↔	0.079↑	0.054↔	0.054↔	0.000↓	0.054↔	0.054↔
	Precision	0.042	0.042↔	0.029↓	0.042↔	0.079↑	0.042↔	0.042↔	0.000↓	0.042↔	0.042↔
	Macro F-measure	0.047	0.047↔	0.034↓	0.047↔	0.079↑	0.047↔	0.047↔	0.000↓	0.047↔	0.047↔
QALD-5 Test dataset	Recall	0.250	0.250↔	0.330↑	0.250↔	0.250↔	0.350↑	0.300↑	0.100↓	0.300↑	0.310↑
	Precision	0.300	0.300↔	0.338↑	0.300↔	0.300↔	0.383↑	0.240↓	0.100↓	0.240↓	0.375↑
	Macro F-measure	0.272	0.272↔	0.353↑	0.272↔	0.272↔	0.366↑	0.267↓	0.100↓	0.267↓	0.340↑
QALD-6 Training Dataset	Recall	0.140	0.140↔	0.152↑	0.140↔	0.167↑	0.161↑	0.146↑	0.031↓	0.106↓	0.152↑
	Precision	0.148	0.148↔	0.163↑	0.148↔	0.187↑	0.165↑	0.134↓	0.031↓	0.093↓	0.163↑
	Macro F-measure	0.144	0.144↔	0.157↑	0.144↔	0.176↑	0.163↑	0.140↓	0.031↓	0.099↓	0.158↑
QALD-6 Test dataset	Recall	0.042	0.042↔	0.042↔	0.084↑	0.084↑	0.048↑	0.125↑	0.000↓	0.126↑	0.042↔
	Precision	0.042	0.042↔	0.042↔	0.125↑	0.125↑	0.083↑	0.125↑	0.000↓	0.167↑	0.042↔
	Macro F-measure	0.042	0.042↔	0.042↔	0.100↑	0.100↑	0.061↑	0.125↑	0.000↓	0.143↑	0.042↔
QALD-7 Training Dataset	Recall	0.093	0.117↑	0.104↑	0.104↑	0.120↑	0.104↑	0.115↑	0.016↓	0.097↑	0.099↑
	Precision	0.101	0.125↑	0.112↑	0.130↑	0.143↑	0.118↑	0.113↑	0.025↓	0.106↑	0.108↑
	Macro F-measure	0.097	0.121↑	0.108↑	0.115↑	0.131↑	0.111↑	0.114↑	0.019↓	0.101↑	0.103↑
# of increased performance			$^2/_7$	$^4/_7$	$^3/_7$		$^5/_7$	$^2/_7$	$^0/_7$	$^2/_7$	$^3/_7$
# of increased & maintained performance			$^7/_7$	$^5/_7$	$^7/_7$		$^7/_7$	$^4/_7$	$^1/_7$	$^5/_7$	$^7/_7$

# of increased performance shows, for each feature, the fraction of datasets with a better F-measure compared to HAWK. # of increased & maintained performance shows the fraction of datasets where the feature improves or maintains the F-measure. For instance, Phrase Combination has led to improve the F-measure of four datasets out of seven tested datasets. It also helps in improving and maintaining the F-measure of five datasets.

Table 5.5. Comparison of HAWK_R features' performance using optimal ranking

QALD Challenge	Measures	HAWK	Question Type Classification	Phrase Combination	Linguistic Pruning	Semantic Annotation Verification	Text Search	Cardinality Limit Identification	Semantic Pruning	Answer Verification	Triple-based Ranking
QALD-4 Training	Recall	0.243	0.243↓	0.209↑	0.059↔	0.063↑	0.079↑	0.051↓	0.004↓	0.057↓	0.059↔
	Precision	0.176	0.209↓	0.147↑	0.090↔	0.130↑	0.110↑	0.087↓	0.040↓	0.095↑	0.090↔
	Macro F-measure	0.204	0.203↓	0.112↑	0.071↔	0.085↑	0.092↑	0.064↓	0.007↓	0.071↔	0.071↔
QALD-4 Test Dataset	Recall	0.100	0.200↑	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔
	Precision	0.100	0.200↑	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔
	Macro F-measure	0.100	0.200↑	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔	0.100↔
QALD-5 Training	Recall	0.151	0.151↓	0.079↑	0.054↔	0.054↔	0.054↔	0.054↔	0.000↓	0.054↔	0.054↔
	Precision	0.112	0.112↓	0.079↑	0.042↔	0.042↔	0.042↔	0.042↔	0.000↓	0.042↔	0.042↔
	Macro F-measure	0.128	0.128↓	0.079↑	0.047↔	0.047↔	0.047↔	0.047↔	0.000↓	0.047↔	0.047↔
QALD-5 Test Dataset	Recall	0.350	0.330↑	0.250↔	0.350↑	0.250↔	0.250↔	0.300↑	0.100↓	0.300↑	0.310↑
	Precision	0.383	0.400↑	0.300↔	0.383↑	0.300↔	0.300↔	0.240↓	0.100↓	0.240↓	0.375↑
	Macro F-measure	0.366	0.373↑	0.272↔	0.366↑	0.272↔	0.272↔	0.267↓	0.100↓	0.267↓	0.340↑
QALD-6 Training	Recall	0.256	0.256↑	0.167↑	0.161↑	0.140↔	0.140↔	0.146↑	0.031↓	0.106↓	0.152↑
	Precision	0.254	0.258↑	0.187↑	0.165↑	0.148↔	0.148↔	0.134↓	0.031↓	0.093↓	0.163↑
	Macro F-measure	0.255	0.257↑	0.176↑	0.163↑	0.144↔	0.144↔	0.140↓	0.031↓	0.099↓	0.158↑
QALD-6 Test Dataset	Recall	0.150	0.192↔	0.084↑	0.048↑	0.084↑	0.042↔	0.125↑	0.000↓	0.126↑	0.042↔
	Precision	0.179	0.221↔	0.125↑	0.083↑	0.125↑	0.042↔	0.125↑	0.000↓	0.167↑	0.042↔
	Macro F-measure	0.164	0.205↔	0.100↑	0.061↑	0.100↑	0.042↔	0.125↑	0.000↓	0.143↑	0.042↔
QALD-7 Training	Recall	0.224	0.234↑	0.120↑	0.104↑	0.104↑	0.117↑	0.115↑	0.016↓	0.097↑	0.099↑
	Precision	0.221	0.232↑	0.143↑	0.118↑	0.130↑	0.125↑	0.113↑	0.025↓	0.106↑	0.108↑
	Macro F-measure	0.263	0.233↑	0.131↑	0.111↑	0.115↑	0.121↑	0.114↑	0.019↓	0.101↑	0.103↑

Among the modified and new features, five features have led to better and / or maintained the F-measure on all the tested datasets using overlap-based ranking. Question type classification in step 1, Linguistic pruning in step 5, Semantic annotation verification in step 6, Text search in step 7 and Triple-based ranking in Step 10. On the opposite side, our new Semantic pruning

technique has led to lower the performance on most datasets. The graph in Figure 5.14 demonstrates the F-measure of HAWK_R modified features on different datasets in comparison to HAWK.

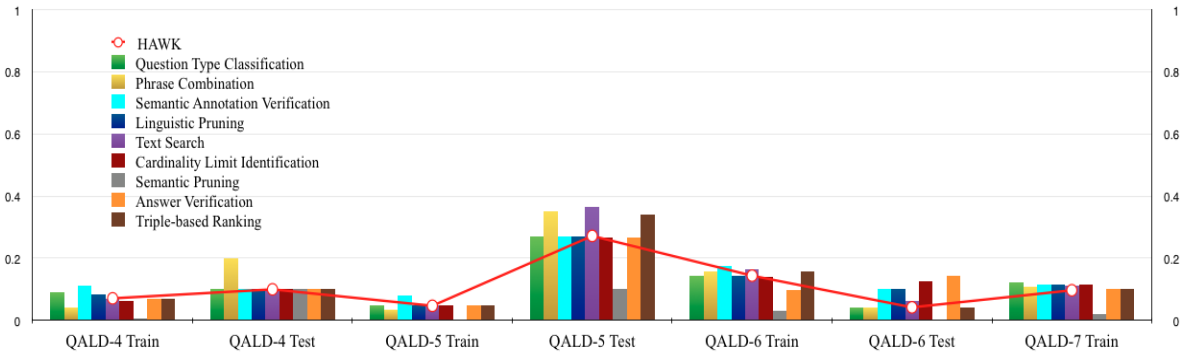


Figure 5.14. Performance (Macro F-measure) of HAWK_R features over different datasets

Based on these results, we combined the best features to implement HAWK_R. A modification and/or new feature is considered valuable if it helps in increasing or maintaining the performance (in terms of macro F-measure) of the system over at least half of our datasets (four datasets). Seven features among the nine tested features maintained the previous HAWK performance on the datasets. Among these features, several contributed to an increase of performance in at least two datasets. Semantic annotation verification helped in improving the performance on five data sets. Phrase combination and Text search enhanced the performance on four datasets. Linguistic pruning and Triple-based ranking raised the performance on three datasets. Question type classification, Cardinality limit identification and Answer verification improved the results on two datasets.

5.11 HAWK_R EVALUATION

We used the generic question answering architecture (proposed in Chapter 2) as a guideline for our HAWK_R system. Figure 5.15 shows the architectural overview of HAWK_R where we add new modules: Semantic annotation verification, answer verification and triple-based ranking, together with modifications in several existing modules as previously explained.

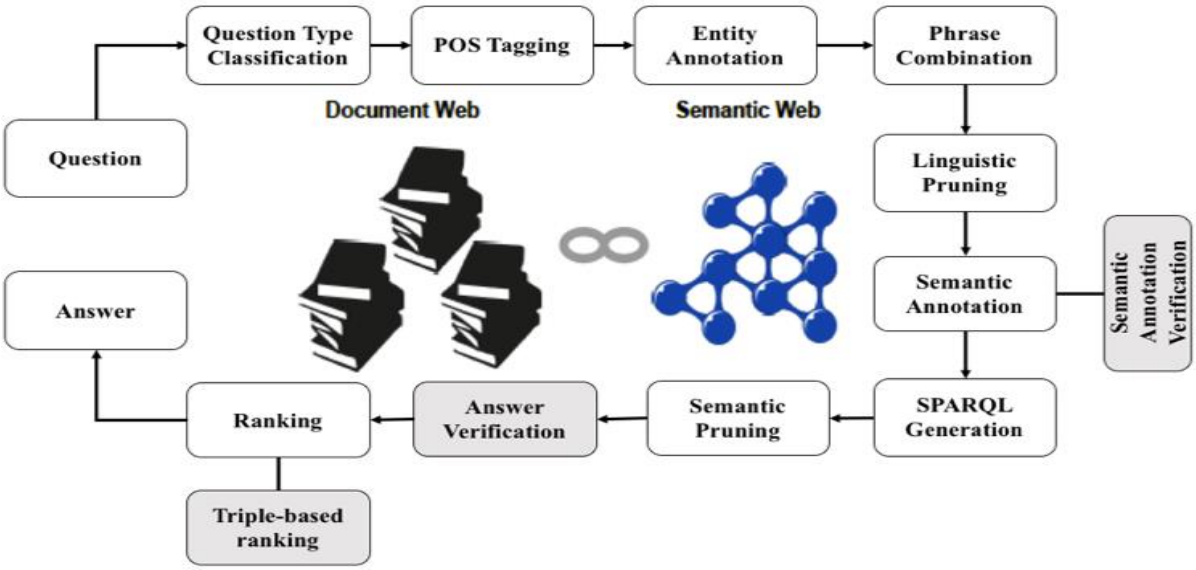


Figure 5.15. Architectural overview of HAWK_R. Adapted from (Usbeck et al., 2015)

We used several datasets to train and test our system HAWK_R. Training datasets are not used in a machine learning experiment, but rather are used for analyzing the datasets, identifying the correct classifications and patterns and evaluating the effectiveness of each proposed enhancement. Then, after finding the correct features and combination of modifications, we evaluate the efficiency of HAWK_R over test datasets, which differ from the ones used for training.

5.11.1 Training Datasets Performance

5.11.1.1 Overall Performance

HAWK_R was tested over several datasets. We used seven datasets to "train" our system HAWK_R. The training datasets include QALD-4 to QALD-6 training and test datasets and QALD-7 training dataset.

The graph in Figure 5.16 displays the performance of HAWK_R using triple-based ranking after combining the best features on our training datasets in comparison to HAWK, which uses overlap-based ranking.

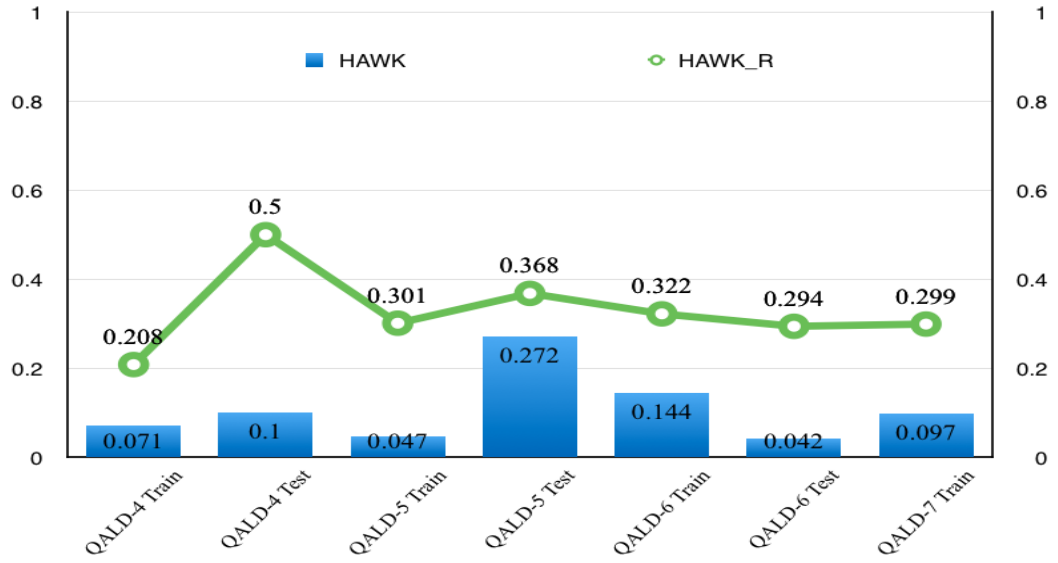


Figure 5.16. Performance (Macro F-measure) of HAWK versus HAWK_R over training datasets

5.11.1.2 Performance of Ranking Techniques

The influence of the applied combined changes on our training datasets in comparison with HAWK is shown in Table 5.6, Table 5.7 and Table 5.8. Table 5.6 displays the performance of HAWK_R using overlap-based ranking, which returns the most frequent answer as the final answer. Table 5.7 shows the performance of the system using optimal ranking, which compares answers with the gold standard.

Table 5.6. Performance of HAWK_R versus HAWK using overlap-based ranking (Combined Features)

QALD	***Total*** # of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	4	21	0.059	0.090	0.071	0.081
HAWK_R QALD-4 Training Dataset	25	22	1	5	19	0.148↑	0.193↑	0.167↑	0.191↑
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	10	4	0	6	0.400↑	0.400↑	0.400↑	0.400↑
HAWK QALD-5 Training Dataset	40	34	0	3	37	0.054	0.042	0.047	0.056
HAWK_R QALD-5 Training Dataset	40	37	8	3	29	0.255↑	0.246↑	0.250↑	0.271↑
HAWK QALD-5 Test dataset	10	9	2	1	7	0.250	0.300	0.272	0.303
HAWK_R QALD-5 Test dataset	10	9	3	1	6	0.400↑	0.340↑	0.367↑	0.408↑

HAWK QALD-6 Training Dataset	49	41	5	4	40	0.140	0.148	0.144	0.172
HAWK_R QALD-6 Training Dataset	49	45	12	4	33	0.313↑	0.290↑	0.301↑	0.328↑
HAWK QALD-6 Test dataset	24	16	1	0	23	0.042	0.042	0.042	0.063
HAWK_R QALD-6 Test dataset	24	18	5	2	17	0.221↑	0.238↑	0.229↑	0.295↑
HAWK QALD-7 Training Dataset	102	85	6	7	89	0.093	0.101	0.097	0.116
HAWK_R QALD-7 Training Dataset	102	92	22	9	71	0.259 ↑	0.269 ↑	0.264 ↑	0.287↑

Table 5.7. Performance of HAWK_R versus HAWK using optimal ranking (Combined Features)

QALD	***Total*** # of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	9	16	0.243	0.176	0.204	0.232
HAWK_R QALD-4 Training Dataset	25	22	8	9	8	0.588↑	0.557↑	0.572↑	0.650↑
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	10	6	0	4	0.600↑	0.600↑	0.600↑	0.600↑
HAWK QALD-5 Training Dataset	40	34	1	8	31	0.151	0.112	0.128	0.151
HAWK_R QALD-5 Training Dataset	40	37	16	7	17	0.530↑	0.500↑	0.515↑	0.556↑
HAWK QALD-5 Test dataset	10	9	2	2	6	0.350	0.383	0.366	0.406
HAWK_R QALD-5 Test dataset	10	9	4	2	4	0.600↑	0.503↑	0.547↑	0.608 ↑
HAWK QALD-6 Training Dataset	49	41	6	9	33	0.256	0.254	0.255	0.305
HAWK_R QALD-6 Training Dataset	49	45	22	10	17	0.594↑	0.562↑	0.578↑	0.629↑
HAWK QALD-6 Test dataset	24	16	2	3	19	0.150	0.179	0.164	0.245
HAWK_R QALD-6 Test dataset	24	18	9	4	11	0.483↑	0.480↑	0.481↑	0.644↑
HAWK QALD-7 Training Dataset	102	85	10	19	73	0.224	0.205	0.214	0.257
HAWK_R QALD-7 Training Dataset	102	92	45	18	39	0.578↑	0.553↑	0.565↑	0.64↑

We also evaluated the performance of our proposed triple-based ranking. Table 5.8 shows the performance of HAWK_R using our proposed triple-based ranking, which uses the query confidence score.

Table 5.8. Performance of HAWK_R versus HAWK using triple-based ranking (Combined Features)

QALD	***Total*** # of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	4	4	21	0.059	0.090	0.071	0.081
HAWK_R QALD-4 Training Dataset	25	22	2	5	18	0.188↑	0.233↑	0.208↑	0.214↑
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	10	5	0	5	0.500↑	0500↑	0.500↑	0.500↑
HAWK QALD-5 Training Dataset	40	34	0	3	37	0.054	0.042	0.047	0.056
HAWK_R QALD-5 Training Dataset	40	37	10	3	27	0.305↑	0.296↑	0.301↑	0.325↑
HAWK QALD-5 Test dataset	10	9	2	2	6	0.310	0.375	0.340	0.377
HAWK_R QALD-5 Test dataset	10	9	3	1	6	0.400↑	0.340↑	0.368↑	0.408↑
HAWK QALD-6 Training Dataset	49	41	5	5	39	0.152	0.163	0.158	0.188
HAWK_R QALD-6 Training Dataset	49	45	13	4	32	0.333↑	0.311↑	0.322↑	0.350↑
HAWK QALD-6 Test dataset	24	16	1	0	23	0.042	0.042	0.042	0.063
HAWK_R QALD-6 Test dataset	24	18	4	4	16	0.304↑	0.285↑	0.294↑	0.406↑
HAWK QALD-7 Training Dataset	102	85	6	8	88	0.099	0.108	0.103	0.124
HAWK_R QALD-7 Training Dataset	102	92	23	10	69	0.298↑	0.300↑	0.299↑	0.330↑

Figure 5.17 shows the performance of HAWK_R in comparison to HAWK using several ranking techniques: the new triple-based ranking and the two original ranking methods overlap-based ranking and optimal ranking. Although our new ranking method does not always give the correct answer, it performs better than overlap-based ranking.

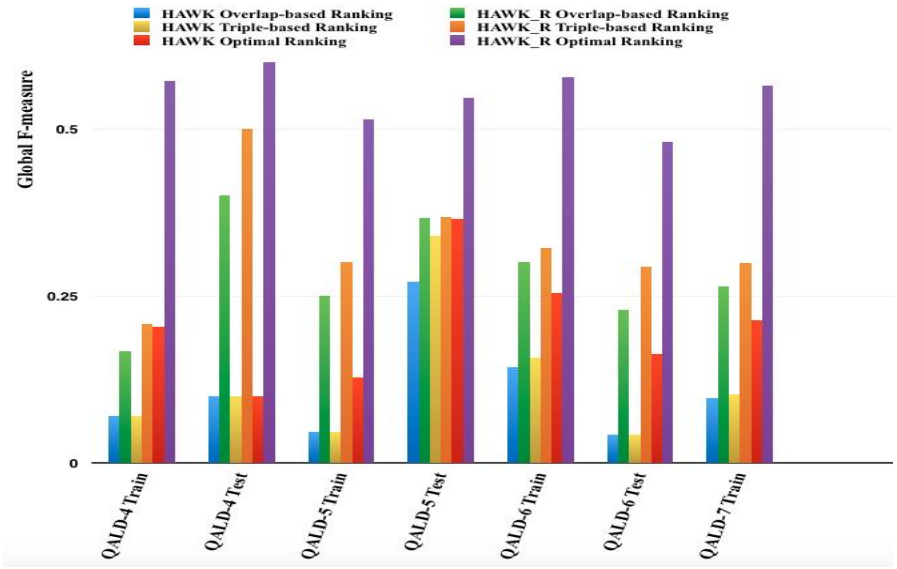


Figure 5.17. Performance (Macro F-measure) of HAWK_R using different ranking techniques over training datasets (Combined Features)

5.11.2 Test Datasets Performance

5.11.2.1 Overall Performance

To test the final version ofHAWK_R after combining only the effective features, we used QALD-7 test dataset and QALD-8 training dataset. Figure 5.18 shows the performance of HAWK_R with triple-based ranking in comparison to HAWK with overlap-based ranking over our test datasets.

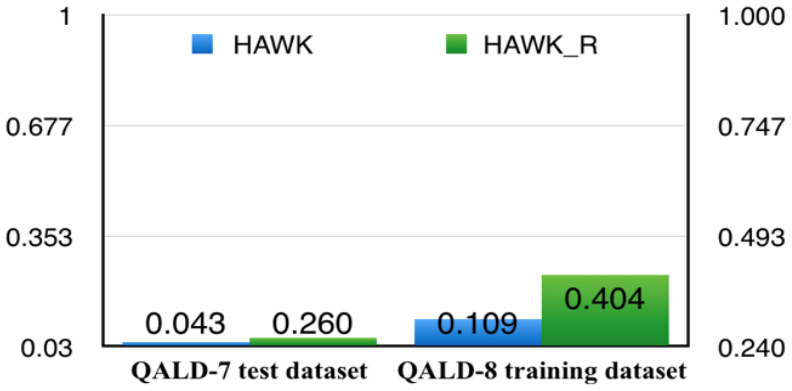


Figure 5.18. Performance (Macro F-measure) of HAWK versus HAWK_R over our test datasets

5.11.2.2 Performance of Ranking Techniques

The influence of the applied combined changes on our test datasets in comparison with HAWK is shown in Table 5.9 and Figure 5.19. The table shows the performance of HAWK_R using the three ranking techniques: overlap-based ranking, triple-based ranking and optimal ranking.

Table 5.9. Performance of HAWK_R versus HAWK using different ranking techniques (Combined Features)

QALD	**System**	***Ranking technique***	***Total*** # of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
QALD-7 Test dataset	HAWK	overlap-based	50	26	2	1	47	0.043	0.043	0.043	0.083
	HAWK_R	overlap-based	50	36	13	0	37	0.240↑	0.240↑	0.240↑	0.333↑
	HAWK_R	triple-based	50	36	13	0	37	0.260↑	0.260↑	0.260↑	0.361↑
	HAWK	optimal	50	26	2	1	47	0.043	0.043	0.043	0.083
	HAWK_R	optimal	102	93	35	12	55	0.401↑	0.408↑	0.404↑	0.440↑
QALD-8 Training Dataset	HAWK	overlap-based	102	86	6	7	89	0.105	0.112	0.109	0.126
	HAWK_R	overlap-based	102	93	25	9	68	0.280 ↑	0.320 ↑	0.299 ↑	0.307↑
	HAWK_R	triple-based	50	36	22	1	27	0.442↑	0.442↑	0.442↑	0.614↑
	HAWK	optimal	102	86	11	18	73	0.237	0.232	0.234	0.278
	HAWK_R	optimal	102	93	52	17	33	0.623↑	0.612↑	0.617↑	0.683↑

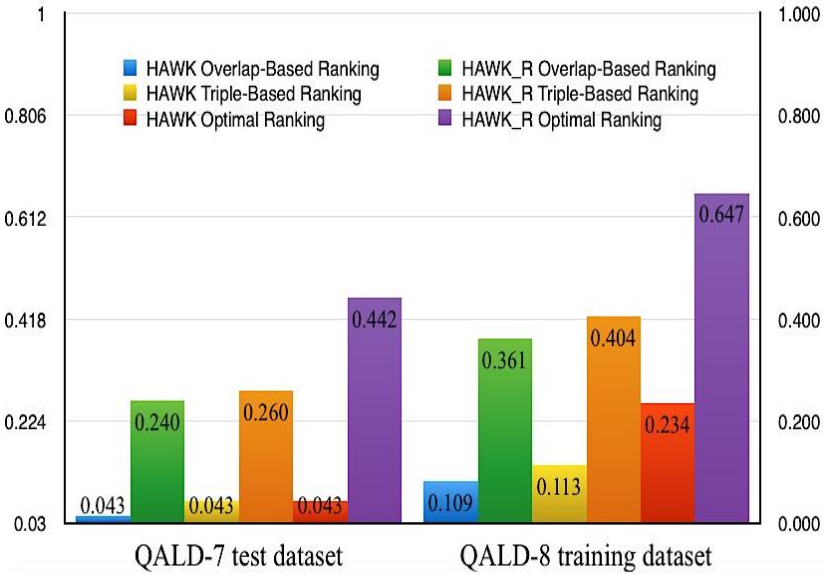


Figure 5.19. Performance (Macro F-measure) of HAWK_R over our test datasets using different ranking techniques (Combined Features)

As we can notice, we were able to increase the performance on the two datasets using triple-based ranking and optimal ranking.

CHAPTER 6. CONCLUSION AND FUTURE WORK

6.1 CONCLUSION AND CONTRIBUTIONS

In this thesis, we improved a question answering system called HAWK, which is an open source system for hybrid questions. We analyzed the system and its limitations and proposed solutions to overcome them. Some problems were solved by modifying the techniques used by HAWK, whereas other problems required building new modules. Moreover, we analyzed the system's performance after combining the top features, which led to the system HAWK_R. We proposed several new features that have a positive and noticeable effect on the performance. The macro F-measure raised from 0.109 to 0.404 for HAWK and HAWK_R respectively on the QALD-8 training dataset. The average F-measure of all the nine tested datasets is 0.103 and 0.284 for HAWK and HAWK_R, respectively.

Our methods are not limited to HAWK and can also help increase the performance of other question answering systems. Overall, we proposed an approach to verify the coherence of semantic annotations based on the features of the question (other semantic annotations and question type). Moreover, we used several methods to answer different types of questions such as number, date and year answers that require operations such as counting and text extraction that go beyond triple pattern matching. In addition, we developed new techniques to verify the answer type, and to rank answer sets based on the query's confidence score.

6.2 SHORTCOMINGS AND FUTURE WORK

Although the work proposed in this thesis has a promising impact on the performance, it is an exploratory work that still has a number of limitations. The main considered limitations can be classified into five categories.

The first limitation is related to the fact that free text is looked up in DBpedia abstracts only, in some datasets, which has limited the scope of text search. Using Wikipedia as a resource of unstructured data would expand the search scope and hence raise the possibility of finding the correct answer, which sometimes does not exist in the abstracts.

Another shortcoming is the failure to identify some named entities that are required to answer the question. For instance, in the question "*What is the largest city in the county in which*

Faulkner spent most of his life?", DBpedia spotlight is not able to find the resource *dbr:William_Faulkner*.

In some cases, we are not able to predict the answer type correctly. For instance, HAWK_R classifies the question "*Under which pseudonym did Charles Dickens publish some of his books?"* as a resource (instead of String) because it has a similar pattern to other resource questions.

Another issue is query pruning. In some cases, the pruning rules remove correct queries. However, more relaxed pruning rules lead to a huge number of queries that have to be fired and to a larger verification of the answer sets, which is time-consuming.

The last main issue is related to ranking the answers. Although our ranking technique performs better than HAWK's overlap-based ranking, it still does not always return the correct answer. In some cases, the correct answer has a lower score compared to other wrong answers in the answer set and thus it is ignored. This points to the need for a further exploration of the ranking step.

The gold standards used in our experiments also have some limitations as they include some incorrect answers, predicates and cardinalities. In some questions, the provided answer is not correct. For instance, the question "*How many awards achieved the creator of the world's most famous equation?*", the gold standard answer is 6 but *dbr:Albert_Einstein* received 7 awards which are:

dbr:Nobel_Prize_in_Physics, dbr:Barnard_Medal_for_Meritorious_Service_to_Science, dbr:Matteucci_Medal, dbr:Gold_Medal_of_the_Royal_Astronomical_Society, dbr:Copley_Medal, dbr:Max_Planck_Medal, dbr:Fellow_of_the_Royal_Society.

In other questions, wrong predicates are used in the gold standard query and hence the answer is wrong. For instance, in the question "*Are there man-made lakes in Australia that are deeper than 100 meters?*", the gold standard query uses *dbo:depth*. However, none of the 16 "*man-made lakes in Australia*" currently uses the predicate *dbo:depth*. Since inappropriate properties are used sometimes (most likely due to DBpedia updates), the answer is necessarily wrong or empty. Another problem is that some answers' cardinalities are not always correct. For

instance, in the question "*Which buildings owned by the crown overlook the North Sea?*" we expect more than one answer from the clue "*buildings*". However, the question has only one answer in the gold standard, and this leads to lower the precision.

Additionally, the DBpedia knowledge base contains incorrect entity types and resource redirects that complicate the question answering task. In fact, to ensure that an answer is correct, its type is compared with the clue type. In some cases, the DBpedia resource has a wrong type. For example, the answer of the question "*Gaborone is the capital of which country member of the African Union?*" is *dbr:Botswana* and the clue is *country*. However, the entity type of *dbr:Botswana* is *dbo:Musical_Artist*.

Overcoming the above-mentioned issues would be necessary to increase the performance of HAWK_R in particular and hybrid question answering systems overall.

REFERENCES

Aggarwal, N., & Buitelaar, P. (2012). A system description of natural language query over DBpedia. *In: C. Unger, P. Cimiano, V. Lopez, E. Motta, P. Buitelaar, R. Cyganiak (eds.): Proceedings of Interacting with Linked Data (ILD 2012), Workshop co-located with the 9th Extended Semantic Web Conference, Heraklion, Greece, 28-05-2012* (Vol. 913, pp. 96-99).

Antoniou, G., Grobelnik, M., Simperl, E., Parsia B., Plexousakis, D., De Leenheer, P., & Pan, J.Z. (Eds.). (2011). *The Semanic Web: Research and Applications: 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29 - June 2, 2011. Proceedings (Vol. 6644). Springer.*

Baudiš, P. (2015, September). YodaQA: a modular question answering system pipeline. In *POSTER 2015-19th International Student Conference on Electrical Engineering* (pp. 1156-1165).

Beaumont, R., Grau, B., & Ligozat, A.L. (2015). SemGraphQA@QALD-5: LIMSI participation at QALD-5@CLEF. In *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015, CEUR Workshop Proceedings (Vol. 1391).*

Cabrio, E., Cimiano, P., Lopez, V., Ngomo, AC.N., Unger, C., & Walter, S. (2012). 3ed open challenge on question answering over linked data (QALD-3). *Proceedings of the first Workshop on Bio-Medical Semantic Indexing and Question Answering, a Post-Conference Workshop of Conference and Labs of the Evaluation Forum 2013 (CLEF 2013), Valencia, Spain, September 27th, 2013. CEUR Workshop Proceedings* (Vol. 1094).

Cabrio, E., Cojan, J., Aprosio, A.P., Magnini, B., Lavelli, A., & Gandon, F. (2012, November). QAKiS: an open domain QA system based on relational patterns. In *International Semantic Web Conference, ISWC 2012* (Vol. 7649, pp. 9-12).

Campbell, L.M., & MacNeill, S. (2010). The semantic web, linked and open data. *A Briefing Paper. Einsehbar unter http://wiki. cetis. ac. uk/images/1/1a/The_Semantic_Web. pdf*.

Coyle, K. (2012). *Linked data tools: connecting on the Web*. American Library Association.

Damljanovic, D., Agatonovic, M., & Cunningham, H. (2010, May). Identification of the Question Focus: Combining Syntactic Analysis and Ontology-based Lookup through the User Interaction. In *LREC 2010 (pp. 361-368).*

Damljanovic, D., Agatonovic, M., & Cunningham, H. (2010, May). Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In *Extended Semantic Web Conference* (pp. 106-120). Springer, Berlin, Heidelberg.

Damljanovic, D., Agatonovic, M., & Cunningham, H. (2011, May). FREyA: An interactive way of querying Linked Data using natural language. In *Extended Semantic Web Conference* (pp. 125-138). Springer, Berlin, Heidelberg.

De Marneffe, M. C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., & Manning, C. D. (2014, May). Universal Stanford dependencies: A cross-linguistic typology. In *LREC* (Vol. 14, pp. 4585-4592).

Diefenbach, D., Lopez, V., Singh, K., & Maret, P. (2017). Core techniques of question answering systems over knowledge bases: a survey. *Knowledge and Information Systems*, (pp. 1-41).

Dima, C. (2013, September). Intui2: A prototype system for question answering over linked data. In Forner et al., (Eds.): *Working Notes for CLEF Conference 2013, (*Vol. 1179*).*

Dima, C. (2014, September). Answering natural language questions with Intui3. In Cappellato et al., (Eds.): *Working Notes for CLEF Conference 2014*, (Vol. 1180, pp. 1201-1211).

Dubien, S. (2005). *Question answering using document tagging and question classification* (Doctoral dissertation, Lethbridge, Alta.: University of Lethbridge, Faculty of Arts and Science, 2005).

Fensel, D., Facca, F.M., Simperl, E., & Toma, I. (2011). *Semantic web services.* Springer Science & Business Media 2011, ISBN 978-3-642-19192-3, pp. I-XI, 1-357.

Ferré, S. (2013, September). squall2sparql: a Translator from Controlled English to Full SPARQL 1.1. In Work. *Multilingual Question Answering over Linked Data (QALD-3).* In Forner et al., (Eds.): *Working Notes for CLEF Conference 2013, (*Vol. 1179*).*

Giannone, C., Bellomaria, V. & Basili, R. (2013, September). A HMM-based Approach to Question Answering against Linked Data. In Forner et al., (Eds.): *Working Notes for CLEF Conference 2013, (*Vol. 1179*).*

Guyonvarch, J., & Ferré, S. (2013). Scalewelis: a scalable query-based faceted search elena work. *Multilingual Question Answering over Linked Data (QALD-3*. In Forner et al., (Eds.): *Working Notes for CLEF Conference 2013, (*Vol. 1179*).*

Hakimov, S., Unger, C., Walter, S. & Cimiano, P. (2015, June). Applying semantic parsing to question answering over linked data: Addressing the lexical gap. In *International Conference on Applications of Natural Language to Information Systems* (pp. 103-109). Springer, Cham.

He, S., Liu, S., Chen, Y., Zhou, G., Liu, K., & Zhao, J. (2013). CASIA@QALD-3: A Question Answering System over Linked Data. In Forner et al., (Eds.): *Working Notes for CLEF Conference 2013, (*Vol. 1179*).*

He, S., Zhang, Y., Liu, K., & Zhao, J. (2014). CASIA@V2: A MLN-based Question Answering System over Linked Data. In Cappellato et al., (Eds.): *Working Notes for CLEF Conference* 2014, (Vol. 1180, pp. 1249-1259).

Höffner, K., Walter, S., Marx, E., Usbeck, R., Lehmann, J., & Ngomo, A. N. (2017). Survey on challenges of question answering in the semantic web. *Semantic Web*, (Vol. 8, No. 6, pp. 895-920).

Klein, D., & Manning, C.D. (2003). Accurate Unlexicalized Parsing. In *Proceedings of the 41st annual meeting of the association for computational linguistics* (pp. 423-430).

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, Sö., & Bizer, C. (2015). DBpedia - a large-scale,

multilingual Knowledge base extracted from Wikipedia. *Semantic Web*, (Vol. 6, No. 2, pp. 167-195).

Nakashole, N., Weikum, G., & Suchanek, F. (2012, July). PATTY: a taxonomy of relational patterns with semantic types. *In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 1135-1145). Association for Computational Linguistics.

Niwattanakul, S., Singthongchai, J., Naenudorn, E., & Wanapu, S. (2013). Using of Jaccard coefficient for keywords similarity. In *Proceedings of the International MultiConference of Engineers and Computer Scientists* (Vol. 1, No. 6, pp. 380-384).

Park, S., Kwon, S., Kim, B., & Lee, G.G. (2015). ISOFT at QALD-5: Hybrid Question Answering System over Linked Data and Text Data. In Cappellato et al., (Eds.): *Working Notes for CLEF Conference 2015, (Vol. 1391).*

Park, S., Shim, H., & Lee, G.G. (2014). ISOFT at QALD-4: Semantic similarity-based question answering system over linked data. In Cappellato et al., (Eds.): *Working Notes for CLEF Conference 2014, (Vol. 1180, pp. 1236-1248).*

Pradel, C., Peyet, G., Haemmerle, O., & Hernandez, N. (2013). SWIP at QALD-3: results, criticisms and lesson learned. In Forner et al., (Eds.): *Working Notes for CLEF Conference 2013,* (Vol. 1179).

Ruseti, S., Mirea, A., Rebedea, T. & Trausan-Matu, S. (2015). QAnswer-Enhanced Entity Matching for Question Answering over Linked Data. In Cappellato et al., (Eds.): *Working Notes for CLEF Conference 2015,* (Vol. 1391).

Sack, H., Blomqvist, E., d'Aquin, M., Ghidini, C., Ponzetto, S.P., & Lange, C. (2016). The Semantic Web: Latest Advances and New Domains: *13th international conference, ESWC 2016, Heraklion, Crete, Greece, May 29--June 2, 2016, Proceedings* (Vol. 9678). Springer.

Taylor, A., Marcus, M., & Santorini, B. (2003). The Penn treebank: an overview. In *Treebanks* (pp. 5-22). Springer, Dordrecht.

Unger, C., Cimiano, P., Lopez, V., & Motta, E. (2011). Proceedings of the 1st Workshop on Question Answering Over Linked Data (QALD-1). *8th Extended Semantic Web Conference*.

Unger, C., Cimiano, P., Lopez, V., Motta, E. Buitelaar, P., & Cyganiak, R. (2012). Interacting with Linked Data (ILD 2012), In *CEUR Workshop Proceedings 2012*, (Vol. 913).

Unger, C., Forascu, C., Lopez, V., Ngomo, A.C.N., Cabrio, E., Cimiano, P., & Walter, S. (2014). 4th Open Challenge on Question Answering over Linked Data (QALD-4). In *Working Notes for CLEF 2014 Conference*, (Vol. 1180, pp. 1172-1180).

Unger, C., Forascu, C., Lopez, V., Ngomo, A.C.N., Cabrio, E., Cimiano, P., & Walter, S. (2015). 5th Open Challenge on Question Answering over Linked Data (QALD-5). In Cappellato et al., (Eds.): *Working Notes for CLEF 2015 Conference*, (Vol. 1391).

Unger, C., Ngomo, A.C.N., & Cabrio, E. (2016, May). 6th open challenge on question answering over linked data (QALD-6). In H. Sack et al. (Eds.): *SemWebEval 2016*, (pp. 171–177).

Usbeck, R., & Ngomo, A.C.N. (2015). HAWK@QALD-5: Trying to answer hybrid questions with various simple ranking techniques. In Cappellato et al., (Eds.): *CLEF (Working Notes) 2015*, (Vol. 1391).

Usbeck, R., Ngomo, A. C. N., Haarmann, B., Krithara, A., Röder, M., & Napolitano, G. (2017, May). 7th Open Challenge on Question Answering over Linked Data (QALD-7). In Dragoni et al. (Eds.): *Semantic Web Evaluation Challenge* (pp. 59-69). Springer, Cham.

Usbeck, R., Ngomo, A.C.N., Buhmann, L., & Unger, C. (2015). HAWK - Hybrid Question Answering over Linked Data. In Gandon et al. (Eds.): *ESWC 2015*, (pp. 353–368).

Veyseh, A.P. (2016). Cross-Lingual Question Answering Using Profile HMM & Unified Semantic Space. In *13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 – June 2, 2016*.

Xu, K., Zhang, S., Feng, Y., & Zhao, D. (2014, September). Xser@ QALD-4: Answering natural language questions via phrasal semantic parsing. In Cappellato et al., (Eds.): *Working Notes for CLEF 2014 Conference* (Vol. 1180, pp. 15-18).

Yahya, M., Berberich, K., Elbassuoni, S., & Weikum, G. (2013, October). Robust question answering over the web of linked data. In *Proceedings of the 22nd ACM International Conference on information & knowledge management* (pp. 1107-1116). ACM.

Zou, L., Huang, R., Wang, H., Yu, J. X., He, W. & Zhao, D. (2014, June). Natural language question answering over RDF: a graph data driven approach. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (pp. 313-324). ACM.

ANNEX

1. An Example of Performance Calculation

Table 0.1 shows how the evaluation measures are computed per question.

Table 0.1. Detailed results per question (QALD-5 Test dataset)

ID	Question	HAWK Answer	Gold Standard Answer	Recall (q)	Precision (q)
1	Where was the "Father of Singapore" born?	*dbr:Aw_Boon_Par*	dbr:Jamaica	0	0
2	Which Secretary of State was significantly involved in the United States' dominance of the Caribbean?	dbr:Ocacia	dbr:James_G._Blaine	0	0
3	Who is the architect of the tallest building in Japan?	dbr:Jun_Mitsui	dbr:Nikken_Sekkei OR César_Pelli	0	0
4	What is the name of the Viennese newspaper founded by the creator of the croissant?	dbr:Evga_S.A dbr:SPC_Group dbr:Paris_Croissant	Die Presse	0	0
5	In which city where Charlie Chaplin's half brothers born?	Queries do not give any answer	dbr:London	0	0
6	Which German mathematicians were members of the von Braun rocket group?	dbr:Oswald_Lange	dbr:Walter_Haeussermann dbr:Oswald_Lange	$\frac{1}{2} = 0.5$	1
7	Which writers converted to Islam?	dbr:Abdillahi_Nassir dbr:Vyacheslav_Polosin dbr:Muslim_Khatris dbr:Yusuf_Tabrizi	dbr:Jamilah_Kolocotronis dbr:Jacky_Trevane dbr:Ismael_Urbain dbr:Lev_Nussimbaum	0	0

			dbr:Aukai_Collins		
8	Are there man-made lakes in Australia that are deeper than 100 meters?	true	true	1	1
9	Which movie by the Coen brothers stars John Turturro in the role of a New York City playwright?	dbr:Barton_Fink	dbr:Barton_Fink	1	1
10	Which of the volcanoes that erupted in 1550 is still active?	dbr:Poás_Volcano	dbr:Volcán_Barú	0	0

Table 0.2 illustrates the process of using the recall and precision of each question to compute the Macro F-measure on the specified dataset

Table 0.2. Calculating average of Recall, Precision and F-measure (QALD-5 Test dataset)

Macro Recall	$\frac{0+0+0+0+0+0.5+0+1+1+0}{10} = \frac{2.5}{10} = 0.25$
Macro Precision	$\frac{0+0+0+0+0+1+0+1+1+0}{10} = \frac{3}{10} = 0.3$
Macro F-measure	$\frac{2\ X\ 0.25 X\ 0.3}{0.25+\ 0.3} = 0.27$

2. Results Based on Individual Features

Table 0.3 – Table 0.18 indicate the performance of HAWK_R with the modification of each step over different datasets using overlap-based ranking and optimal ranking.

2.1. Question Type Classification

Table 0.3. Performance of HAWK_R versus HAWK using overlap-based ranking (Question type classification)

QALD	***Total*** # of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	4	21	0.059	0.090	0.071	0.081
HAWK_R QALD-4 Training Dataset	25	22	0	5	20	0.079↑	0.110↑	0.092↑	0.104↑
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143

HAWK_R QALD-4 Test dataset	10	7	1	0	9	0.100↔	0.100↔	0.100↔	0.143↔
HAWK QALD-5 Training Dataset	40	34	0	3	37	0.054	0.042	0.047	0.056
HAWK_R QALD-5 Training Dataset	40	34	0	3	37	0.054↔	0.042↔	0.047↔	0.056↔
HAWK QALD-5 Test dataset	10	9	2	1	7	0.250	0.300	0.272	0.303
HAWK_R QALD-5 Test dataset	10	9	2	1	7	0.250↔	0.300↔	0.272↔	0.303↔
HAWK QALD-6 Training Dataset	49	41	5	4	40	0.140	0.148	0.144	0.172
HAWK_R QALD-6 Training Dataset	49	41	5	4	40	0.140↔	0.148↔	0.144↔	0.172↔
HAWK QALD-6 Test dataset	24	16	1	0	23	0.042	0.042	0.042	0.063
HAWK_R QALD-6 Test dataset	24	16	1	0	23	0.042↔	0.042↔	0.042↔	0.063↔
HAWK QALD-7 Training Dataset	102	85	6	7	89	0.093	0.101	0.097	0.116
HAWK_R QALD-7 Training Dataset	102	101	8	8	86	0.117↑	0.125↑	0.121↑	0.122↑

Table 0.4. Performance of HAWK_R versus HAWK using optimal ranking (Question type classification)

QALD	***Total #*** of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	9	16	0.243	0.176	0.204	0.232
HAWK_R QALD-4 Training Dataset	25	22	0	9	16	0.243↔	0.176↔	0.204↔	0.232↔
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	7	1	0	9	0.100↔	0.100↔	0.100↔	0.143↔
HAWK QALD-5 Training Dataset	40	34	1	8	31	0.151	0.112	0.128	0.151
HAWK_R QALD-5 Training Dataset	40	34	2	8	30	0.151↔	0.112↔	0.128↔	0.151↔
HAWK QALD-5 Test dataset	10	9	2	2	6	0.350	0.383	0.366	0.406
HAWK_R QALD-5 Test dataset	10	9	2	2	6	0.350↔	0.383↔	0.366↔	0.406↔
HAWK QALD-6 Training Dataset	49	41	7	9	33	0.256	0.254	0.255	0.305
HAWK_R QALD-6 Training Dataset	49	41	7	9	33	0.256↔	0.254↔	0.255↔	0.305↔
HAWK QALD-6 Test dataset	24	16	2	3	19	0.150	0.179	0.164	0.245
HAWK_R QALD-6 Test dataset	24	16	2	3	19	0.150↔	0.179↔	0.164↔	0.245↔
HAWK QALD-7 Training Dataset	102	85	10	19	73	0.224	0.205	0.214	0.257
HAWK_R QALD-7 Training Dataset	102	101	14	17	71	0.249↑	0.234↑	0.241↑	0.243↑

2.2. Phrase Combination

Table 0.5. Performance of HAWK_R versus HAWK using overlap-based ranking (Phrase combination)

QALD	***Total*** # of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	4	21	0.059	0.090	0.071	0.081
HAWK_R QALD-4 Training Dataset	25	24	0	2	23	0.031↓	0.067↓	0.042↓	0.044↓
HAWK QALD-4 Test dataset	10	7	0	1	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	10	2	0	8	0.200↑	0.200↑	0.200↑	0.200↑
HAWK QALD-5 Training Dataset	40	34	0	3	37	0.054	0.042	0.047	0.056
HAWK_R QALD-5 Training Dataset	40	37	1	2	37	0.042↓	0.029↓	0.034↓	0.037↓
HAWK QALD-5 Test dataset	10	9	2	1	7	0.250	0.300	0.272	0.303
HAWK_R QALD-5 Test dataset	10	9	2	2	6	0.330↑	0.380↑	0.353↑	0.392↑
HAWK QALD-6 Training Dataset	49	41	5	4	40	0.140	0.148	0.144	0.172
HAWK_R QALD-6 Training Dataset	49	44	5	4	40	0.152↑	0.163↑	0.157↑	0.175↑
HAWK QALD-6 Test dataset	24	16	1	0	23	0.042	0.042	0.042	0.063
HAWK_R QALD-6 Test dataset	24	19	1	0	23	0.042↔	0.042↔	0.042↔	0.053↓
HAWK QALD-7 Training Dataset	102	85	6	7	89	0.093	0.101	0.097	0.116
HAWK_R QALD-7 Training Dataset	102	93	6	7	89	0.104↑	0.112↑	0.108↑	0.118↑

Table 0.6. Performance of HAWK_R versus HAWK using optimal ranking (Phrase combination)

QALD	***Total*** # of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	9	16	0.243	0.176	0.204	0.232
HAWK_R QALD-4 Training Dataset	25	24	1	8	16	0.243↔	0.176↔	0.204↔	0.212↓
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	10	2	0	8	0.200↑	0.200↑	0.200↑	0.200↑
HAWK QALD-5 Training Dataset	40	34	1	8	31	0.151	0.112	0.128	0.151
HAWK_R QALD-5 Training Dataset	40	37	1	8	31	0.151↔	0.112↔	0.128↔	0.139↓
HAWK QALD-5 Test dataset	10	9	2	2	6	0.350	0.383	0.366	0.406
HAWK_R QALD-5 Test dataset	10	9	3	1	6	0.350↔	0.400↑	0.373↑	0.415↑
HAWK QALD-6 Training Dataset	49	41	7	9	33	0.256	0.254	0.255	0.305
HAWK_R QALD-6 Training Dataset	49	44	8	8	33	0.256↔	0.258↑	0.257↑	0.286↓
HAWK QALD-6 Test dataset	24	16	2	3	19	0.150	0.179	0.164	0.245
HAWK_R QALD-6 Test dataset	24	16	3	3	18	0.192↑	0.221↑	0.205↑	0.308↑
HAWK QALD-7 Training Dataset	102	85	10	19	73	0.224	0.205	0.214	0.257

HAWK_R QALD-7 Training Dataset	102	102	14	16	72	0.234↑	0.232↑	0.233↑	0.233↓

2.3. Linguistic Pruning

Table 0.7. Performance of HAWK_R versus HAWK using overlap-based ranking (Linguistic pruning)

QALD	***Total* #** of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	4	21	0.059	0.090	0.071	0.081
HAWK_R QALD-4 Training Dataset	25	24	0	5	20	0.063↑	0.130↑	0.085↑	0.089↑
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	7	1	0	9	0.100↔	0.100↔	0.100↔	0.143↔
HAWK QALD-5 Training Dataset	40	34	0	3	37	0.054	0.042	0.047	0.056
HAWK_R QALD-5 Training Dataset	40	35	0	3	37	0.054↔	0.042↔	0.047↔	0.054↓
HAWK QALD-5 Test dataset	10	9	2	1	7	0.250	0.300	0.272	0.303
HAWK_R QALD-5 Test dataset	10	9	2	1	7	0.250↔	0.300↔	0.272↔	0.303↔
HAWK QALD-6 Training Dataset	49	41	5	4	40	0.140	0.148	0.144	0.172
HAWK_R QALD-6 Training Dataset	49	42	4	4	41	0.140↔	0.141↓	0.144↔	0.168↓
HAWK QALD-6 Test dataset	24	16	1	0	23	0.042	0.042	0.042	0.063
HAWK_R QALD-6 Test dataset	24	17	2	1	21	0.084↑	0.125↑	0.100↑	0.142↑
HAWK QALD-7 Training Dataset	102	85	6	7	89	0.093	0.101	0.097	0.116
HAWK_R QALD-7 Training Dataset	102	97	6	9	87	0.104↑	0.130↑	0.115↑	0.121↑

Table 0.8. Performance of Modified HAWK_R versus HAWK using optimal ranking (Linguistic pruning)

QALD	***Total* #** of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	9	16	0.243	0.176	0.204	0.232
HAWK_R QALD-4 Training Dataset	25	24	1	8	16	0.243↔	0.176↔	0.204↔	0.212↔
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	7	1	0	9	0.100↔	0.100↔	0.100↔	0.143↔
HAWK QALD-5 Training Dataset	40	34	1	8	31	0.151	0.112	0.128	0.151
HAWK_R QALD-5 Training Dataset	40	35	1	8	31	0.151↔	0.112↔	0.128↔	0.147↓
HAWK QALD-5 Test dataset	10	9	2	2	6	0.350	0.383	0.366	0.406
HAWK_R QALD-5 Test dataset	10	9	2	2	6	0.350↔	0.383↔	0.366↔	0.406↔

HAWK QALD-6 Training Dataset	49	41	7	9	33	0.256	0.254	0.255	0.305
HAWK_R QALD-6 Training Dataset	49	42	7	9	33	0.256↔	0.254↔	0.255↔	0.297↓
HAWK QALD-6 Test dataset	24	16	2	3	19	0.150	0.179	0.164	0.245
HAWK_R QALD-6 Test dataset	24	17	3	3	18	0.192↑	0.221↑	0.205↑	0.290↑
HAWK QALD-7 Training Dataset	102	85	10	19	73	0.224	0.205	0.214	0.257
HAWK_R QALD-7 Training Dataset	102	97	13	17	72	0.239↑	0.230↑	0.232↑	0.244↓

2.4. Semantic Annotation

Table 0.9. Performance of HAWK_R versus HAWK using overlap-based ranking (Semantic annotation)

QALD	***Total #*** of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	4	21	0.059	0.090	0.071	0.081
HAWK_R QALD-4 Training Dataset	25	23	1	3	21	0.091↑	0.147↑	0.112↑	0.122↑
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	7	1	0	9	0.100↔	0.100↔	0.100↔	0.143
HAWK QALD-5 Training Dataset	40	34	0	3	37	0.054	0.042	0.047	0.056
HAWK_R QALD-5 Training Dataset	40	37	1	3	36	0.079↑	0.079↑	0.079↑	0.086↑
HAWK QALD-5 Test dataset	10	9	2	1	7	0.250	0.300	0.272	0.303
HAWK_R QALD-5 Test dataset	10	10	2	1	7	0.250↔	0.300↔	0.272↔	0.272↓
HAWK QALD-6 Training Dataset	49	41	5	4	40	0.140	0.148	0.144	0.172
HAWK_R QALD-6 Training Dataset	49	43	6	4	39	0.167↑	0.187↑	0.176↑	0.201↑
HAWK QALD-6 Test dataset	24	16	1	0	23	0.042	0.042	0.042	0.063
HAWK_R QALD-6 Test dataset	24	17	2	1	21	0.084↑	0.125↑	0.100↑	0.142↑
HAWK QALD-7 Training Dataset	102	85	6	7	89	0.093	0.101	0.097	0.116
HAWK_R QALD-7 Training Dataset	102	90	9	7	86	0.120↑	0.144↑	0.131↑	0.149↑

Table 0.10. Performance of HAWK_R versus HAWK using optimal ranking (Semantic annotation)

QALD	***Total* # of questions**	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	9	16	0.243	0.176	0.204	0.232
HAWK_R QALD-4 Training Dataset	25	23	4	9	12	0.401↑	0.336↑	0.366↑	0.398↑
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	7	1	0	9	0.100↔	0.100↔	0.100↔	0.143
HAWK QALD-5 Training Dataset	40	34	1	8	31	0.151	0.112	0.128	0.151
HAWK_R QALD-5 Training Dataset	40	37	5	8	27	0.251↑	0.212↑	0.230↑	0.248↑
HAWK QALD-5 Test dataset	10	9	2	2	6	0.350	0.383	0.366	0.406
HAWK_R QALD-5 Test dataset	10	10	2	2	6	0.350↔	0.383↔	0.366↔	0.366↓
HAWK QALD-6 Training Dataset	49	41	7	9	33	0.256	0.254	0.255	0.305
HAWK_R QALD-6 Training Dataset	49	43	12	9	28	0.358↑	0.356↑	0.357↑	0.407↑
HAWK QALD-6 Test dataset	24	16	2	3	19	0.150	0.179	0.164	0.245
HAWK_R QALD-6 Test dataset	24	17	4	3	17	0.234↑	0.261↑	0.246↑	0.348↑
HAWK QALD-7 Training Dataset	102	85	10	19	73	0.224	0.205	0.214	0.257
HAWK_R QALD-7 Training Dataset	102	90	19	17	66	0.293↑	0.289↑	0.291↑	0.326↑

2.4 Cardinality Limit Identification

Table 0.11 indicates the performance of HAWK_R when using different cardinality numbers over QALD-7 training dataset.

Table 0.11. Performance of HAWK_R over QALD-7 training dataset using overlap-based ranking (Cardinality limit identification)

QALD	***Limit number***	***Total* # of questions**	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-7 Training Dataset	12	102	85	6	7	89	0.093	0.101	0.097	0.116
HAWK_R QALD-7 Training Dataset	22	102	85	5	5	92	0.086↓	0.095↓	0.090↓	0.108↓
	100			6	8	88	0.107↑	0.109↑	0.108↑	0.129↑
	NoLimit			6	8	88	0.107↑	0.108↑	0.107↑	0.129↑

Table 0.12. Performance of HAWK_R versus HAWK using overlap-based ranking (Cardinality limit identification)

QALD	***Total* #** of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	4	21	0.059	0.090	0.071	0.081
HAWK_R QALD-4 Training Dataset	25	22	0	3	22	0.051↓	0.087↓	0.064↓	0.073↓
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	7	1	0	9	0.100↔	0.100↔	0.100↔	0.143↔
HAWK QALD-5 Training Dataset	40	34	0	3	37	0.054	0.042	0.047	0.056
HAWK_R QALD-5 Training Dataset	40	33	0	3	37	0.054↔	0.042↔	0.047↔	0.057↑
HAWK QALD-5 Test dataset	10	9	2	1	7	0.250	0.300	0.272	0.303
HAWK_R QALD-5 Test dataset	10	10	2	1	9	0.300↑	0.240↓	0.267↓	0.266↓
HAWK QALD-6 Training Dataset	49	41	5	4	40	0.140	0.148	0.144	0.172
HAWK_R QALD-6 Training Dataset	49	42	5	3	42	0.146↑	0.134↓	0.140↓	0.163↓
HAWK QALD-6 Test dataset	24	16	1	0	23	0.042	0.042	0.042	0.063
HAWK_R QALD-6 Test dataset	24	16	3	0	21	0.125↑	0.125↑	0.125↑	0.186↑
HAWK QALD-7 Training Dataset	102	85	6	7	89	0.093	0.101	0.097	0.116
HAWK_R QALD-7 Training Dataset	102	85	8	6	88	0.115↑	0.113↑	0.114↑	0.208↑

Table 0.13. Performance of HAWK_R versus HAWK using optimal ranking (Cardinality limit identification)

QALD	***Total* #** of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	9	16	0.243	0.176	0.204	0.232
HAWK_R QALD-4 Training Dataset	25	22	1	8	16	0.260↑	0.210↑	0.232↑	0.264↑
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	7	1	0	9	0.100↔	0.100↔	0.100↔	0.143↔
HAWK QALD-5 Training Dataset	40	34	1	8	31	0.151	0.112	0.128	0.151
HAWK_R QALD-5 Training Dataset	40	33	2	6	32	0.139↓	0.116↑	0.128↔	0.155↑
HAWK QALD-5 Test dataset	10	9	2	2	6	0.350	0.383	0.366	0.406
HAWK_R QALD-5 Test dataset	10	10	2	2	6	0.350↔	0.383↔	0.366↔	0.366↓
HAWK QALD-6 Training Dataset	49	41	7	9	33	0.256	0.254	0.255	0.305
HAWK_R QALD-6 Training Dataset	49	42	6	9	34	0.256	0.222↓	0.238↓	0.272↓
HAWK QALD-6 Test dataset	24	16	2	3	19	0.150	0.179	0.164	0.245
HAWK_R QALD-6 Test dataset	24	16	4	3	17	0.239↑	0.257↑	0.248↑	0.372↑
HAWK QALD-7 Training Dataset	102	85	10	19	73	0.224	0.205	0.214	0.257

HAWK_R QALD-7 Training Dataset	102	85	14	16	72	0.245↑	0.228↑	0.236↑	0.284↑

2.6. Text Search

Table 0.14. Performance of HAWK_R per versus HAWK using overlap-based ranking (Text search)

QALD	***Total* #** of questions	# of ***processed*** questions	# of ***correctly*** answered	# of ***partially*** answered questions	# of ***incorrectly*** answered	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	4	21	0.059	0.090	0.071	0.081
HAWK_R QALD-4 Training Dataset	25	22	0	4	21	0.059↔	0.090↔	0.071↔	0.081↔
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	7	1	0	9	0.100↔	0.100↔	0.100↔	0.143↔
HAWK QALD-5 Training Dataset	40	34	0	3	37	0.054	0.042	0.047	0.056
HAWK_R QALD-5 Training Dataset	40	33	0	3	37	0.054↔	0.042↔	0.047↔	0.057↓
HAWK QALD-5 Test dataset	10	9	2	1	7	0.250	0.300	0.272	0.303
HAWK_R QALD-5 Test dataset	10	9	2	2	6	0.350↑	0.383↑	0.366↑	0.406↑
HAWK QALD-6 Training Dataset	49	41	5	4	40	0.140	0.148	0.144	0.172
HAWK_R QALD-6 Training Dataset	49	43	5	5	39	0.161↑	0.165↑	0.163↑	0.186↑
HAWK QALD-6 Test dataset	24	16	1	0	23	0.042	0.042	0.042	0.063
HAWK_R QALD-6 Test dataset	24	16	1	1	22	0.048↑	0.083↑	0.061↑	0.091↑
HAWK QALD-7 Training Dataset	102	85	6	7	89	0.093	0.101	0.097	0.116
HAWK_R QALD-7 Training Dataset	102	88	6	9	87	0.104↑	0.118↑	0.111↑	0.135↑

Table 0.15. Performance of HAWK_R per versus HAWK using optimal ranking (Text search)

QALD	***Total* #** of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	9	16	0.243	0.176	0.204	0.232
HAWK_R QALD-4 Training Dataset	25	22	1	1	16	0.282↑	0.216↑	0.245↑	0.278↑
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	7	2	0	8	0.200↑	0.200↑	0.200↑	0.286↑
HAWK QALD-5 Training Dataset	40	34	1	8	31	0.151	0.112	0.128	0.151
HAWK_R QALD-5 Training Dataset	40	33	2	8	30	0.176↑	0.137↑	0.154↑	0.188↑
HAWK QALD-5 Test dataset	10	9	2	2	6	0.350	0.383	0.366	0.406
HAWK_R QALD-5 Test dataset	10	9	2	2	6	0.350↔	0.383↔	0.366↔	0.406↔
HAWK QALD-6 Training Dataset	49	41	7	9	33	0.256	0.254	0.255	0.305

HAWK_R QALD-6 Training Dataset	49	43	8	9	32	0.286↑	0.262↑	0.274↑	0.312↑
HAWK QALD-6 Test dataset	24	16	2	3	19	0.150	0.179	0.164	0.245
HAWK_R QALD-6 Test dataset	24	16	3	3	19	0.190↑	0.220↑	0.205↑	0.307↑
HAWK QALD-7 Training Dataset	102	85	10	19	73	0.224	0.205	0.214	0.257
HAWK_R QALD-7 Training Dataset	102	88	14	17	71	0.260↑	0.245↑	0.252↑	0.292↑

2.7. Semantic Pruning

Table 0.16. Performance of HAWK_R versus HAWK using overlap-based ranking (Semantic pruning)

QALD	***Total* #** of questions	# of ***processed*** questions	# of ***correctly*** answered	# of ***partially*** answered questions	# of ***incorrectly*** answered	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	4	21	0.059	0.090	0.071	0.081
HAWK_R QALD-4 Training Dataset	25	5	0	1	24	0.004↓	0.040↓	0.007↓	0.036↓
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	7	1	0	9	0.100↔	0.100↔	0.100↔	0.143↔
HAWK QALD-5 Training Dataset	40	34	0	3	37	0.054	0.042	0.047	0.056
HAWK_R QALD-5 Training Dataset	40	34	0	0	40	0.000↓	0.000↓	0.000↓	0.000↓
HAWK QALD-5 Test dataset	10	9	2	1	7	0.250	0.300	0.272	0.303
HAWK_R QALD-5 Test dataset	10	10	1	0	9	0.100	0.100↓	0.100↓	0.100↓
HAWK QALD-6 Training Dataset	49	41	5	4	40	0.140	0.148	0.144	0.172
HAWK_R QALD-6 Training Dataset	49	41	1	1	48	0.031↓	0.031↓	0.031↓	0.037↓
HAWK QALD-6 Test dataset	24	16	1	0	23	0.042	0.042	0.042	0.063
HAWK_R QALD-6 Test dataset	24	16	0	0	24	0.000↓	0.000↓	0.000↓	0.000↓
HAWK QALD-7 Training Dataset	102	85	6	7	89	0.093	0.101	0.097	0.116
HAWK_R QALD-7 Training Dataset	102	10	1	1	100	0.016↓	0.025↓	0.019↓	0.195↑

Table 0.17. Performance of HAWK_R versus HAWK using optimal ranking (Semantic pruning)

QALD	***Total* #** of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	9	16	0.243	0.176	0.204	0.232
HAWK_R QALD-4 Training Dataset	25	5	0	1	24	0.032↓	0.036↓	0.034↓	0.169↓
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	7	1	0	9	0.100↔	0.100↔	0.100↔	0.143↔
HAWK QALD-5 Training Dataset	40	34	1	8	31	0.151	0.112	0.128	0.151

HAWK_R QALD-5 Training Dataset	40	34	0	2	38	0.027↓	0.025↓	0.026↓	0.031↓
HAWK QALD-5 Test dataset	10	9	2	2	6	0.350	0.383	0.366	0.406
HAWK_R QALD-5 Test dataset	10	10	1	0	9	0.100↓	0.100↓	0.100↓	0.100↓
HAWK QALD-6 Training Dataset	49	41	7	9	33	0.256	0.254	0.255	0.305
HAWK_R QALD-6 Training Dataset	49	41	1	2	46	0.043↓	0.039↓	0.041↓	0.049↓
HAWK QALD-6 Test dataset	24	16	2	3	19	0.150	0.179	0.164	0.245
HAWK_R QALD-6 Test dataset	24	16	0	0	24	0.000↓	0.000↓	0.000↓	0.000↓
HAWK QALD-7 Training Dataset	102	85	10	19	73	0.224	0.205	0.214	0.257
HAWK_R QALD-7 Training Dataset	102	10	1	4	97	0.038↓	0.032↓	0.035↓	0.357↑

2.8.Answer Verification

Table 0.18. Performance of HAWK_R versus HAWK using overlap-based ranking (Answer verification)

QALD	***Total #*** of questions	# of ***processed*** questions	# of ***correctly*** answered	# of ***partially*** answered questions	# of ***incorrectly*** answered	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	4	21	0.059	0.090	0.071	0.081
HAWK_R QALD-4 Training Dataset	25	22	4	4	21	0.057↓	0.095↑	0.071↔	0.080↓
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	7	1	0	9	0.100↔	0.100↔	0.100↔	0.143↔
HAWK QALD-5 Training Dataset	40	34	0	3	37	0.054	0.042	0.047	0.056
HAWK_R QALD-5 Training Dataset	40	34	0	3	37	0.054↔	0.042↔	0.047↔	0.056↔
HAWK QALD-5 Test dataset	10	9	2	1	7	0.250	0.300	0.272	0.303
HAWK_R QALD-5 Test dataset	10	9	2	1	7	0.300↑	0.240↓	0.267↓	0.296↓
HAWK QALD-6 Training Dataset	49	41	5	4	40	0.140	0.148	0.144	0.172
HAWK_R QALD-6 Training Dataset	49	42	3	3	43	0.106↓	0.093↓	0.099↓	0.116↓
HAWK QALD-6 Test dataset	24	16	1	0	23	0.042	0.042	0.042	0.063
HAWK_R QALD-6 Test dataset	24	15	3	1	20	0.126↑	0.167↑	0.143↑	0.229↑
HAWK QALD-7 Training Dataset	102	85	6	7	89	0.093	0.101	0.097	0.116
HAWK_R QALD-7 Training Dataset	102	85	6	8	88	0.097↑	0.106↑	0.101↑	0.122↑

Table 0.19. Performance of HAWK_R versus HAWK using optimal ranking (Answer verification)

QALD	***Total #*** of questions	# of ***processed*** questions	# of ***correctly*** answered questions	# of ***partially*** answered questions	# of ***incorrectly*** answered questions	***Macro Recall***	***Macro Precision***	***Macro F-measure***	***Micro F-measure***
HAWK QALD-4 Training Dataset	25	22	0	9	16	0.243	0.176	0.204	0.232
HAWK_R QALD-4 Training Dataset	25	22	2	7	16	0.252↑	0.213↑	0.231↑	0.262↑
HAWK QALD-4 Test dataset	10	7	1	0	9	0.100	0.100	0.100	0.143
HAWK_R QALD-4 Test dataset	10	7	1	0	9	0.100↔	0.100↔	0.100↔	0.143↔
HAWK QALD-5 Training Dataset	40	34	1	8	31	0.151	0.112	0.128	0.151
HAWK_R QALD-5 Training Dataset	40	34	4	6	30	0.174↑	0.174↑	0.174↑	0.205↑
HAWK QALD-5 Test dataset	10	9	2	2	6	0.350	0.383	0.366	0.406
HAWK_R QALD-5 Test dataset	10	9	2	2	6	0.350↔	0.383↔	0.366↔	0.409↑
HAWK QALD-6 Training Dataset	49	41	6	9	33	0.256	0.254	0.255	0.305
HAWK_R QALD-6 Training Dataset	49	42	8	9	32	0.285↑	0.267↑	0.276↑	0.321↑
HAWK QALD-6 Test dataset	24	16	2	3	19	0.150	0.179	0.164	0.245
HAWK_R QALD-6 Test dataset	24	15	4	3	17	0.236↑	0.255↑	0.245↑	0.392↑
HAWK QALD-7 Training Dataset	102	85	10	19	73	0.224	0.205	0.214	0.257
HAWK_R QALD-7 Training Dataset	102	85	16	15	71	0.250↑	0.241↑	0.246↑	0.301↑