Dublin City University at QA@CLEF 2008

Sisay Fissaha Adafre Josef van Genabith*
National Center for Language Technology
School of Computing, DCU
IBM CAS Dublin*
sadafre,josef@computing.dcu.ie

Abstract

We describe our participation in Multilingual Question Answering at CLEF 2008 using German and English as our source and target languages respectively. The system was built using UIMA (Unstructured Information Management Architechture) as underlying framework.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software; H.3.7 Digital Libraries; H.2.3 [**Database Managment**]: Languages—*Query Languages*

General Terms

Measurement, Performance, Experimentation

Keywords

Question Answering, Information Extraction, Importance Ranking

1 Introduction

This is our first participation in the Multilingual Question Answering Track of CLEF. We took part in the bilingual CLEFQA task (German-English) where German is the source language and English the target language. Since the system was originally designed for English, we need to translate the German questions into English. We used the BableFish online translation system to translate the questions. Due to time constraints, no provision has been made to handle errors introduced by the translation system. This in turn has affected the resulting performance of our system since the system was designed assuming syntactically correct questions as input. Furthermore, the system is targeted at *Factoid* and *Definition* questions, and it does not cater for *List* questions.

QA systems, in the context of TREC and CLEF evaluation forums, generally consist of online methods that generate answers to questions automatically by directly analysing the text corpus. Systems also make use of external resources in the form of Gazetteers or precompiled Tables which are obtained through offline mining of large text corpora or the web. Although it has been shown that outputs of offline mining methods can be used to improve QA results, our focus in designing the current system is on testing our online methods which are based on information extraction methods. Our system does not make use of precompiled tables or Gazetteers. Like most systems, our system uses Web snippets to rerank candidate answers extracted from the document collections. In addition to the Web snippets, WordNet is used as lexical resource in the system. Typical QA systems employ various Natural Language Processing (NLP) and Machine Learning

(ML) tools, a set of heuristics and different lexical resources. Seamless integration of the various components is one of the major challenges of QA system development. In order to facilitate our development process, we used the Unstructured Information Management Architecture (UIMA) as our underlying framework [16].

In the remainder of this paper, we will describe our system. Section 2 provides a detailed description of the system. It briefly summarises the UIMA framework (Section 2.1), and then presents a componentwise description of the system that deals with *Factoid* questions. Section 2.8 summarises the treatment of Definition questions. Section 3 presents the result of the experimental evaluation. Finally, Section 4 presents some concluding remarks.

2 System Description

We adopted the traditional architecture in building our system. Our question answering system consists of the following core components: Question Analysis, Passage Retrieval, Sentence Analysis and Answer Selection [7]. Question analysis mainly involves question classification and query generation. We split the documents into passages and indexed them. The passage retrieval component takes queries and retrieves the relevant passages, which are likely to contain an answer. As passages are not the expected response units, they need to be further analysed to extract more focused answers to user questions. The sentence analysis components apply different Machine Learning and Natural Language Processing techniques, such as Named Entity Recognition, Parsing (CFG and Dependency) and Pattern Matching to the sentences, in order to extract candidate answers from them. Finally, the answer selection component takes the output of the different components and generates a ranked list of candidate answers [13].

Each of these components employs various tools, and a set of heuristic rules. In order to ease the development process, we used UIMA (Unstructured Information Management Architecture). UIMA provides a number of benefits. First, it provides a common representation mechanism of the result of the different analysis components. It also facilitates the integration of different analysis tools. In the next section, we summarise the features of UIMA that are important for the development of our QA System.

2.1 The UIMA Framework

UIMA is a framework and a software development kit for developing applications that analyse and elicit relevant knowledge from unstructured information. Such applications, for example, can be used to extract structured data from unstructured text [16]. UIMA facilitates integration of typical natural language processing tasks, such as tokenization, sentence detection, named-entity recognition, POS-tagging and chunking, parsing, etc, for building complex applications such as question answering. In the setting of Multilingual CLEF 2008, Question Answering largely involves processing of large unstructured textual information (the document collection) to find answers to questions. UIMA takes care of much of the low level text analysis details and allows one to focus on the actual problem.

UIMA offers different application development options. We adopted the Collection Processing Architecture as it provides the basic building blocks for processing large document collections. As shown in Figure 1, the collection processing architecture consists of three main components: Collection Reader, Aggregate Analysis Engine, and Consumers. The Collection Reader iterates through the document collection and converts each document into UIMA internal representation, i.e. Common Analysis Structure (CAS). The CAS is passed to the Aggregate Analysis Engine, which is composed of multiple analysis components (Analysis Engine) which carry out the actual analysis of the document. The Consumers take the output of the Aggregate Analysis Engine and make it available in a form suitable for processing by external applications or for storage. Common Analysis Structure provides a common representation of the objects that UIMA analyses. Analysis Engines contain annotators that form an important component of the Collection Processing Engine (CPE). Results of the analysis are recorded in the form of annotations which are associated with

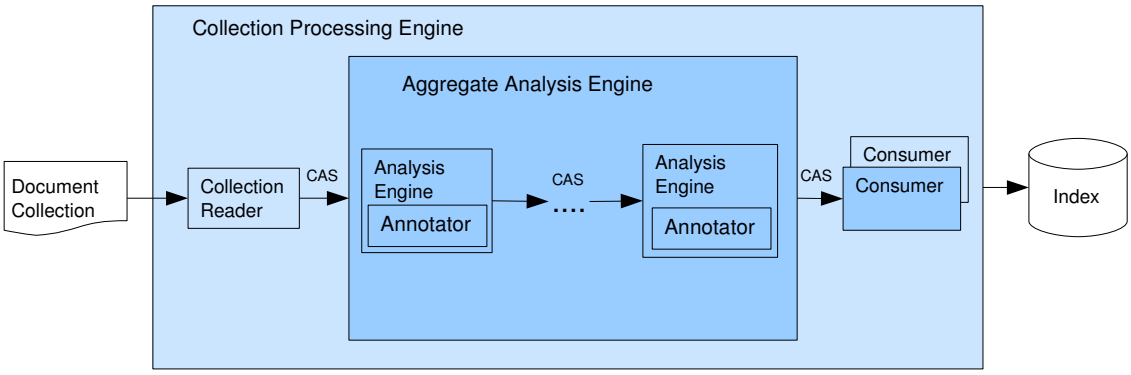

Figure 1: UIMA Architecture

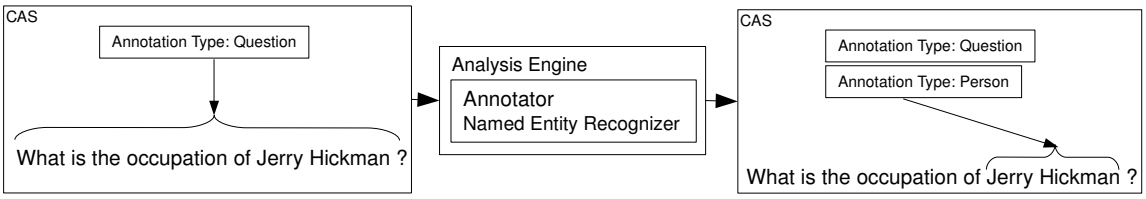

Figure 2: Example Annotator

a region in the object being analysed. For textual data, the annotation is recorded as character offset from the beginning of the document. The CAS provides interfaces for storing and querying annotations over the text document. CAS may contain several annotations where each annotation is identified by the associated type [16].

Figure 2 shows an example UIMA based application containing one analysis engine which recognises named-entities. The input document is made of a single question string. The input CAS contains a *question* annotation which spans the entire question string. The *Named Entity Annotator* takes this input and adds a *Person* annotation to it. The resulting CAS contains two annotations, i.e. *Question* and *Person* annotations. UIMA allows the composition of multiple annotators into aggregate annotators where each annotator adds further annotations to the input CAS. The final CAS consists of the question string and a set of annotations that are associated with spans in the question string.

2.2 System Overview

The main inputs of the system are the document collection and the questions. Corresponding to each of these inputs, we have collection and question processing components. As shown in Figure 3, the collection processing component carries out preprocessing and indexing of the document collection. The preprocessing step involves splitting documents into sentences, and POS tagging and chunking. Indexing takes care of merging a set of sentences into passages and indexing them using the Lucene retrieval engine [10]. The question processing corresponds to question classification and query generation. The question processing also involves POS tagging and chunking, and parsing of the input questions. The question classifier and query generation component take the linguistically annotated questions and generate question classes, and queries respectively. Finally, the

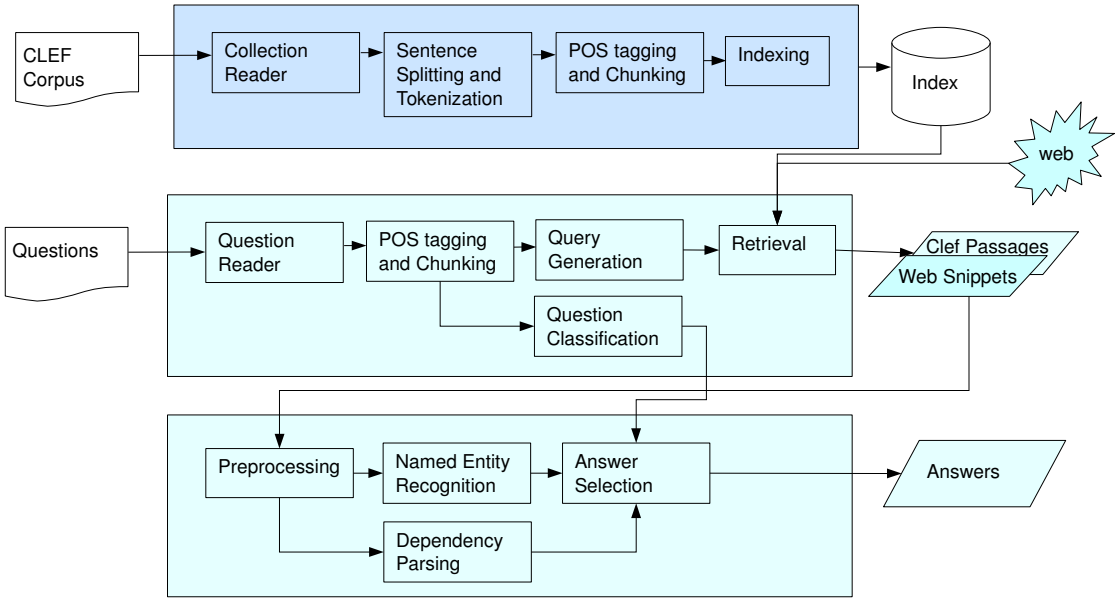

Figure 3: System Architecture

answer selection component, which matches questions to the answers, consists of passage retrieval, sentence analysis, and answer reranking components. In the current system, we used TreeTagger for POS tagging and Chunking, and a treebank-based Lexical Functional Grammar(LFG) parser for dependecy parsing [2, 14].

In the next section, we provide detailed descriptions of each of these components assuming *Factoid* questions as input. Section 2.8 describes the components that deal with *Definition* questions.

2.3 Question Analysis

Question analysis consists of a question classification and a query generation component. We employed both machine-learning and rule-based methods for defining question classes. For the machine learning approach, we trained a classifier using the data set provided by Li and Roth [9]. We used the MinorThird implementation of Conditional Random Fields as our classifier [4]. The classification taxonomy consists of two layers in which the top level consists of 6 major classes (ABBREVIATION, ENTITY, DESCRIPTION, HUMAN, LOCATION and NUMERIC VALUE), and the bottom level consist of 50 classes. The resulting labels are used to define expected answer types for the questions.

The rule based approach uses syntactic clues to identify terms that can be used to refine the expected answer types - *focus* terms. Typical patterns include noun phrase chunks following a *wh* word in the question.

Query generation identifies terms that will be used for retrieving passages. This step uses patterns defined on the output of the POS tagger and Chunker. The query terms consist of noun phrase and verb phrase chuncks, where the focus terms and stop-words are removed.

According to the CLEF guidelines, the questions are organised around a set of topics where each *topic* is associated with a group of questions. However, the *topic* is not explicitly given and must be inferred from the first question or its answer. Therefore, we devised a rule based

method for detecting the *topic* of the first question. The rules make use of both surface patterns and syntactic clues to identify the *topic* of the question. The *topic* of the first question is also appended to the query set of each of the remaining questions.

2.4 Passage Retrieval

We have three kinds of document collections or sources: the CLEF Document Collection, a Wikipedia Corpus, and the Web. We treat each of these sources differently.

CLEF Collection We split the CLEF documents into passages where each passage is composed of 10 consecutive sentences in the document. The sentences in the passages are POS tagged and chunked. The original sentences are indexed and the tagged sentences are stored in a separate field in the index - *CollIndex*.

Wikipedia Collection Similarly, the articles of the Wikipedia corpus are also indexed separately - *WikiIndex*. However, the Wikipedia articles are indexed as a whole and are not split into passages.

The query terms generated by the Question Analysis Module is submitted to both *CollIndex* and *WikiIndex*. The top 100 passages from *CollIndex* and the top 10 articles from *WikiIndex* are retained. Both the passages from the CLEF corpus and Wikipedia articles are split into sentences. Each sentence will be assigned the retrieval status value of the corresponding passage or article as an initial score. The sentences are passed to the sentence analysis module.

Web The system also retrieves web snippets using the same queries. We used Yahoo! APIs for retrieving web snippets. We split the snippets into sentences, and retained only those that contained one or more of the query terms. The sentences are assigned ranks where the top ranking web snippets gets a rank 1, and the second top ranking snippet gets 2 and so on. Finally, sentences are assigned scores as the inverse of their corresponding ranks.

The lists we obtain from the three sources contain scores that are computed quite differently. In order to minimise the effects of the variation in scoring methods, we normalise scores in each list as follows:

$$score_R = \text{score}_{\text{norm}} = \frac{\text{score}_{\text{MAX}} - \text{score}}{\text{score}_{\text{MAX}} - \text{score}_{\text{MIN}}} \quad (1)$$

2.5 Sentence Analysis

We run a named entity recogniser on the sentences retrieved. We trained a CRF based named entity recogniser on the CoNLL Corpus [15]. It recognises the following four major classes: PERSON, ORGANISATION, LOCATION, MISCELLANEOUS. Since there is a mismatch between the classes generated by the question classifier and named entity recognizer, we devised a mapping between the outputs of the two systems. Furthermore, we added additional classes that could reliably be identified using simple pattern matching (such as dates) to the major classes. We have identified the following classes: *Date*, *Person*, *Location*, *Organization*, *Numeric*, *Count* and *Description*. These classes are further qualified by using the bottom levels of the question classification taxonomy, or terms extracted using the rule-based component. For example, *Location* is further qualified by *Country*, *City*, or *State*. The minor classes are used to filter the result set as described in Section 2.7.3. We have one open class which contains all named entities that do not map to any of the above classes.

The sentences are also parsed using a dependency parser. The result is used to extract dependency triples that are used to measure the similarity between Questions and Sentences containing the candidate answer as will be explained in Section 2.7.1.

2.6 Candidate Answer Extraction

We consider two cases when searching for candidate answers to the questions. The first case relates to when the question class can be mapped to one of the predefined classes. In this case, the named entities whose types match one of the broad question classes taken are as candidate answers. For the non-matching case, all noun phrases are extracted from the sentence and are considered candidates. Further type filtering is done on the list as shown in Section 2.7.3. Initially, the named entities receive the retrieval status value of the passage to which they belong. Since several named entities receive the same ranking, we rerank the entities using other evidence of relevance.

2.7 Answer Reranking

We reranked candidate answers based on different sources of evidence, such as syntactic similarity of the sentence with the question, proximity of query terms to the candidate answers, similarity of the semantic type of the candidate answer to the answer type, and centrality of the sentence with respect to a corpus of web snippets retrieved using the query terms extracted from the question. We provide details of each these filtering mechanisms.

2.7.1 Syntactic Similarity

Syntactic structures have been used extensively to find answers to questions. Syntactic structures have been used for reformulation of questions into the corresponding declarative forms containing a missing constituent. The missing constituent is assumed to contain the answer phrase. The resulting pattern is used to search for the sentence containing the answer. The syntactic structures can also be used to measure the similarity between the questions and the sentence containing the candidate answer. Syntax based evidences have been used to rerank candidate answers in a number of QA Systems [6, 8, 12].

In the current system, syntactic similarity is measured in terms of the number of shared dependency relations between the sentence and the question. For this, we parsed both the questions and the sentences using a Lexical Functional Grammar(LFG)-based parser [2] developed at Dublin City University. The system takes the output of a syntactic parser (Charniak parser [3]) and generates an F-Structure, a labeled bilexical dependency graph. The output can also be provided in the form of a set dependency triples. Although the final goal of the project is to make inference based on the F-Structures, the current implementation makes use of the dependency triples to compute similarity between the sentences and the questions. We count how many dependency pairs are shared between the questions and answers, normalise the resulting value and add it to the overall score of the named entities extracted from the sentence.

2.7.2 Term Proximity

This method is based on the assumption that answers are likely to be found in a close proximity to the query terms in a sentence. In other words, if more query terms appear in the vicinity of the candidate answer, the candidate answer is likely to be the true answer and hence receives more weight. We implemented this intuition as follows. For each candidate answer in a sentence, we take a window of 10 terms centred in the candidate answer. We then count how many of the query terms appear in the Window. The final score is obtained by dividing the resulting count by the total number of query terms.

2.7.3 Type Filtering

As mentioned in Section 2.5, our type classification is limited and assigns a significant part of the named entity classes to miscellaneous. On the other hand, the types derived from questions are either specific instances of the major classes we identified or may not be covered by the major

classes. In order to fill the gap, we devised the following methods for computing semantic similarity between the expected answer type and the candidate answer.

Wikipedia Category Wikipedia contains a large set of user defined categories which are assigned to its entries. The method is implemented as a binary feature function. First, we check if the candidate answer has an entry in Wikipedia. If the candidate answer does not match an entry in Wikipedia, we assign a score of 0. If there is a Wikipedia entry corresponding to the answer string, we retrieve the categories associated with the Wikipedia article. We then check if the answer type terms are contained in the category lists. If the answer type terms does not occur in the category list, we assign the candidate term a zero score otherwise we assign a score of 1 to the candidate answer.

WordNet Hierarchy We take both the expected answer type and the candidate answer, and check if they have entries in WordNet. We then check if the expected answer type and the candidate answer stand in the *Hypernym* relation with respect to the WordNet hierarchy. We assign a score which is the inverse of the distance between the two concepts in the hierarchy. For example, if the expected answer type is a direct hypernym of the candidate answer, the candidate answer recieves a score of 1.0, else it will be less than 1.0.

WordNet vs Wikipedia Category This is an extension of *Wikipedia Category* method above. We take the expected answer type, and generate its *WordNet hyponyms* sets (5 levels down the hierarchy). We take the Wikipedia categories of the candidate answer. Finally, we compute the fraction of shared entries between these two sets as a measure of semantic similarity.

2.7.4 Web based evidence

We used the Web in two ways: to find answers, and to rerank candidate answers from the *CLEF Collection* and *Wikipedia Collection*.

Web Answers The Web snippets pass through the same processing pipelines as the snippets (sentences) from the *CLEF Collection* and *Wikipedia Collection*. Since answers must come from the later two collections, the Web answers must be mapped onto the collection (*Answer Projection*). For each candidate answer in the lists obtained from the *CLEF Collection* and *Wikipedia Collection*, we check if there is a matching Web answer in the ranked list of Web answers. If there is, we add the score of the Web answer to the current score of the candidate answer. This assigns more weight to those candidates that are found in the Web answer list.

Web based reranking This type of evidence for sentence importance is based on the assumption that there is a high degree of redundancy among the top web snippets returned for a given query. In order to take advantage of this fact, we create a *reference corpus* consisting of the top 50 ranking Web snippets. This corpus will be used to estimate relevance of the sentences to the questions. This is estimated as a similarity between the target sentence with the web corpus. The target sentence is the sentence that contains the candidate answer. We create a graph with target and reference sentences as nodes and weighted edges indicating similarity between pairs of sentences. In this model, sentences receive evidence of their importance from other sentences and, in turn, they "pass on" their importance, in a recursive manner. Our graph-based ranking method extends a method proposed in [5, 11]: we bring in "weighted support" from the reference corpus. Details of our graph-based ranking approach can be found in [1].

2.7.5 Combining Scores

The overall scores for reranking candidate sentences is computed as a linear combination of the *Retrieval scores*, *Syntactic similarity*, *Term Proximity*,*Type Filtering* and *Web-based evidence*.

Each of these scores have been normalized between 0 and 1 using the formula given 1. The baseline system simply sums these scores without taking into account the relative importance of the different evidences. In the future, we are planning to use machine learning approach for computing the final score using these evidences as features, and a set of questions and answers as training material.

2.8 Definition Questions

The *definition* questions expect short snippets or sentences that provide a concise definition or description of the topic as an answer, unlike *factoids* for which the expected answers are largely named entities. As a result, we adopted a different strategy for definition questions. The system takes the *topic* generated by the question analysis module, and submits it as a query to the retrieval module. The returned passages and Wikipedia articles are split into sentences. Sentences that do not contain the topic are removed from the list. We assign more weight to sentences with copula verb construction with the *topic* as a subject, e.g. TOPIC is ....

Finally, the sentences are reranked using evidences obtained from the web as described in Section 2.7.4.

3 Experimental Evaluation

We submitted two runs for our CLEF participation. The first run is the output of the complete system - *Complete*. The second run is the output of the system without the web-based reranking component - *Partial*.

	Factoid		Definition		Overall	
	Complete	Partial	Complete	Partial	Complete	Partial
Right	8	1	8	0	16	1
Wrong	142	157	16	9	168	195
ineXact	1	1	6	1	7	3
Unsupported	9	1	0	0	9	1

Table 1: Results.

Overall the system returned only 16 *exact* answers, and 25 correct answers counting *unsupported* answers. The web reranking component contributed significantly. The result without the web reranking component is disappointing. This is attributed to a number of problems. The main problem was lack of proper testing due to time constraints. This was compounded by an error introduced by a last minute change. Another major problem is, of course, the scope of our system. As mentioned in Section 1, the system relies primarily on online methods which focused on a restricted class of named entities. Since it is an evolving system, we believe that its coverage will improve by adding more semantic categories.

Effects of Translation Although most of the German questions have been accurately translated into English, there are still some translation errors which affected our results. The errors may range from incorrect word order, missing constituent to non-translated terms, and have affected both the question classification and query generation components. For example, the following translation error resulted in an incorrect question classification - Description or Definition.

- DE: Wie großist Jerod Ward?
- EN: Is Jerod Ward how large?

For the following case, which contains incorrect word order, the query generation component identifies *Mathieu Orfila* as the *Focus* of the sentence, as in *Country* in the question form *Which*

country .... This error in turn propagated to subsequent analysis steps resulting in few candidate answers.

- DE: Wann schrieb Mathieu Orfila sein "Trait des poisons"?
- EN: When Mathieu Orfila wrote its "Trait poisons"?

The last example contains a term that is not translated, i.e. *Geomorphologie*. Since it is the only term in the query generated by the question analsys component, it returned very few snippets.

- DE: Was ist die Geomorphologie?
- EN: What is the Geomorphologie?

4 Conclusion

We presented our QA system adapted for the German-English bilingual CLEFQA task. The system is developed using UIMA (Unstructured Information Management Architecture) as our underlying framework. The exercise showed that UIMA facilitates building QA system in a short period of time. Since this is our first participation, we hope that the results of the system will improve in the future, given adequate time for testing and adapting to the new inputs, i.e. translated inputs.

The system is largely based on Information Extraction methods, with various filtering and reranking steps to pin point the correct answers. It is limited in a number of aspects since it is in its early stages of development. Our future plan is to extend the types of question that can be handled, and improve the methods for those already implemented. Furthermore, we also need to improve on reranking algorithms. We would like to bring in more of the deep NLP methods into the reranking algorithm. Specifically, we would like to extend our dependency triple based scoring method to include the full LFG-based parse output. Finally, computation of the overall score is based on simple linear combination of the individual scores ignoring their relative weights. In the future, we will use a ML based approach for computing the overall score using the individual evidences as features.

5 Acknowledgments

The research reported in this paper was supported by the Science Foundation Ireland under grant number 04/IN/I527.

References

[1] Sisay Fissaha Adafre and Maarten de Rijke. Estimating importance features for fact mining (with a case study in biography mining). In *RIAO*, 2007.

[2] Aoife Cahill, Michael Burke, Martin Forst, Ruth Odonovan, Christian Rohrer, Josef Genabith, and Andy Way. Treebank-based acquisition of multilingual unification grammar resources. *Research on Language and Computation*, 3(2):247–279, July 2005.

[3] Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 132–139, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[4] W. Cohen. Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data. `http://minorthird.sourceforge.net`.

[5] Günes Erkan and Dragomir R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res. (JAIR)*, 22:457–479, 2004.

[6] Boris Katz and Jimmy Lin. Selectively using relations to improve precision in question answering. In *Proceedings of the EACL-2003 Workshop on Natural Language Processing for Question Answering, April 2003*, 2003.

[7] J. Kupiec. Murax: A robust linguistic approach for question-answering using an on-line encyclopedia. In *Proceedings of 16th Annual International ACM/SIGIR Conference.*, 1993.

[8] A. K. Lamjiri, L. Kosseim, and T. Radakrishnan. Comparing the Contribution of Syntactic and Semantic Features in Closed versus Open Domain Question Answering. In *Proceedings of the IEEE International Conference on Semantic Computing (ICSC2007)*, pages 679–685, Irvine, California, USA, 2007.

[9] X. Li and D. Roth. Learning question classifiers, 2002.

[10] Lucene. The Lucene search engine. `http://lucene.apache.org/`.

[11] Rada Mihalcea. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings ACL*, 2004.

[12] D. Molla and M. Gardiner. Answerfinder - question answering by combining lexical. In *In Australasian Language Technology Workshop (ALTW) 2004, Sydney.*, 2004.

[13] Christof. Monz. *From Document Retrieval to Question Answering*. PhD thesis, University of Amsterdam, 2003.

[14] Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, Manchester, UK, 1994.

[15] Erik. F. Tjong Kim Sang. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *COLING-02: Proceedings of the 6th conference on Natural language learning.*, 2002.

[16] UIMA. Unstructured information management architecture. `http://domino.research.ibm.com/comm/research_projects.nsf/pages/uima.index.html`.