Cognitive Computing for Augmented Human Intelligence: Papers from the AAAI-14 Workshop

Using Syntactic Features in Answer Reranking

Jakob Grundström and **Pierre Nugues**
Department of Computer Science
Lund University, Box 118, S-221 00, Lund, Sweden
jakob.grundstrom@gmail.com Pierre.Nugues@cs.lth.se

Abstract

This paper describes a baseline question answering system for Swedish. The system includes modules to carry out the question analysis, hypothesis generation, and reranking of answers. It was trained and evaluated on questions from a data set inspired by the Swedish television quiz show *Kvitt eller Dubbelt*.

We used the Swedish Wikipedia as knowledge source and we show that paragraph retrieval from this corpus gives an acceptable coverage of answers when targeting *Kvitt eller Dubbelt* questions, especially single-word answer questions. Given a question, the hypothesis generation module retrieves a list of paragraphs, ranks them using a vector space model score, and extract a set of candidates.

The question analysis part performs a lexical answer type prediction. To compute a baseline ranking, we sorted answer candidates according to their frequencies in the most relevant paragraphs. The reranker module makes use of information from the previous stages to estimate the correctness of the generated answer candidates as well a grammatical information from a dependency parser. The correctness estimate is then used to re-weight the baseline ranking.

A 5-fold cross-validation showed that the median ranking of the correct candidate went from rank 21 in the baseline version to 10 using the reranker.

Introduction

This paper describes a baseline question answering system for Swedish. The system includes modules for the question analysis, hypothesis generation, and a reranking of answers on which we evaluated the contribution of syntactic features.

The most prominent work in the question answering field is certainly IBM Watson (Gondek et al. 2012), a system that outperformed the human champions of the *Jeopardy!* quiz show. Such a system is extremely appealing as it seems to reach the limits of human intelligence and could find an uncountable number of applications. However, IBM Watson was designed for English and made use the numerous resources available for this language. This makes it difficult to replicate when moving to another language.

In this paper, we show the feasibility of a question answering system for a language with less resources, Swedish in our case. We describe an architecture that copes with such a language and we restricted the search strategies to document search over paragraphs. Its design roughly corresponds to a simplified single path through the Watson pipeline.

We used the Swedish version of Wikipedia (Swedish Wikipedia 2008) as knowledge source and we trained and evaluated our system on questions inspired by the Swedish television quiz show *Kvitt eller Dubbelt – Tiotusenkronorsfrågan* (Thorsvad and Thorsvad 2005; Kvitt eller Dubbelt 2013).

The question analysis part carries out a lexical answer type prediction and is responsible for producing a search query from the question text. The hypothesis generation module uses the Lucene indexer to extract a set of paragraphs. The reranker module makes use of information from the other modules to estimate the correctness of the answer candidates and thus create a new ranking.

Previous Work

In 2011, IBM Watson set a milestone in computing history by beating the two best human champions in a real-time two-game *Jeopardy!* contest (Ferrucci 2012). IBM Watson is, at the time of writing, considered to be the state of the art in question answering.

The IBM Watson system includes, in addition to its question answering core and game play implementation, human-computer interaction components such as: language interpretation, answer formulation, and speech synthesis. In this paper, we focused on the question answering core.

IBM Watson takes full advantage of the *Jeopardy!* format of questions; for instance, in finding the *focus* (the part of the question that refers to the answer) and the lexical answer type (LAT), which indicates the class of the answer in the question analysis. Its question answering architecture, *DeepQA*, is heavily modularized and allows for multiple implementations that can produce alternative results. This creates alternative paths through its pipeline; paths that are independent and can be pursued in parallel (Ferrucci 2012).

Multiple candidate answers can be proposed using different search strategies (Chu-Carroll et al. 2012), for example: document search, passage search, lookup in structured databases with extracted facts (triples), or using knowledge

graphs such as IBM's frame database *Prismatic* (Fan et al. 2012). This also allows for pursuing different interpretations of the question and category. Features and evidence are gathered and analyzed for each answer before the final reranking and selection of the answer candidates. Watson can then use the estimated confidence to decide on its *Jeopardy!* game strategy, e.g. whether or not to answer a question or how much to bet.

Question Corpus

We gathered a corpus of 2310 questions that we transcribed from a card version of the *Kvitt eller Dubbelt – Tiotusenkronorsfrågan* quiz show (Thorsvad and Thorsvad 2005). Out of these questions, 1683 are single-word answers and most of them are nouns. Because of the *Kvitt eller Dubbelt* game play, the questions are divided into 385 sets of six questions of different values printed on 385 different cards. The cards are divided into seven categories, where each card has a name. Each question consists of a *question text*, an *answer*, optionally complemented with an *alternative answer*. We annotated manually answers with one of the eight *answer categories* listed in Table 1. These categories were inspired by Li and Roth (2002).

Table 2 shows three questions from this corpus. Note that some questions have empty fields and that the categorization is partly incomplete. The answer category for question 3, for instance, should be marked as a location. The question text of all the examples in the table consists of a single sentence, as most of the question texts in the corpus. However, some shorter or longer question texts, ranging from one single noun phrase to multiple sentences occasionally occur in the corpus.

Table 1: The answer categories.

misc	entity	description	human
abbrev	action	location	numeric

Since our objective was not to implement the game play but simply the question answering part, we only made use of the following fields: question text, answer, alternative answer, and answer category. We carried out some experiments with the card categories and card names in the question analysis part when formulating queries. Eventually, we only used the question text to build queries.

System Description

Overview

Figure 1 shows an outline of the system architecture and the data flow from the question text to a list of ranked candidate answers. It consists of the following modules, where the input to the system is a full text question:

- The **question analysis** module predicts the answer type using a classifier and is responsible for building a search query.
- The search query is used by the **hypothesis generation** to find the most relevant paragraphs from indexed text resources, in our case the Swedish Wikipedia. Candidates are generated from the retrieved paragraphs. For each candidate, this module stores associated information that might be useful in later stages: parts of speech, dependency graphs, search score, etc.
- The candidates are **merged** when their normalized word forms, lemmas, are equal. The candidate features are merged with independent strategies. Furthermore, this module extracts statistics such as the absolute and relative word frequencies.
- The **reranker** module uses the question text, its syntactic analysis, the predicted answer types with their corresponding estimated probabilities, the merged candidates with their features, and statistics to create a final ranking.

The output from the system is a list of answer candidates ranked by an estimated confidence.

Question Analysis

The question analysis module takes the question text as input and creates a query to extract a set of paragraphs. It applies a part-of-speech tagger (Östling 2013) to lemmatize the question words and predicts its answer category. The module builds an object from these results that is passed to the reranking module. We set aside the card category (i.e. the question topic) from the input as it resulted in a drop in the rank of the first correct answer, when compared to only using the question text.

To predict the answer type, we trained a logistic regression classifier on the question corpus. We implemented the classifier using *LibShortText* (Yu et al. 2013), a library for short text classification based on bags of words and the vector space model. We selected a word and bigram representation, where the word values corresponded to their TF-IDF (term frequency–inverse document frequency) scores. Given a question text, we stored the top two predicted answer types and their associated estimated probabilities in the result object.

We did not use the LibShortText preprocessing options: tokenization, stemming, and stop-word removal that are designed for English. As our text is in Swedish, we normalized the question and we extracted the lemmas from the part-of-speech tagger output instead.

Hypothesis Generation

We implemented the hypothesis generation component using a paragraph-based search. Although other units of text, e.g. article or sentence would also contain question answers, we assumed that paragraphs would yield a better performance because they pack strongly related information. Articles, especially long articles, often contain unrelated information, while single sentences might miss a question context, such as coreferences that span multiple sentences.

We extracted the paragraphs from the Swedish Wikipedia dump using *JSoup* (jsoup: Java HTML Parser 2013) and we

Table 2: Example of card content.

Field	1	2	3
Value	250	5000	5000
Question	*I vilken stad sitter Sveriges regering?*	*Vem är kapten på rymdskeppet Millenium Falcon?*	*I vilket land är Rom huvudstad?*
Translation	'In which city is the Swedish government?'	'Who is the captain of the Millennium Falcon spaceship?'	'Of which country Rome is the capital?'
Answer	Stockholm	Han Solo	Italien
Alt. answer	-	-	-
Answer category	location	human	-
Answer subcategory	-	-	-

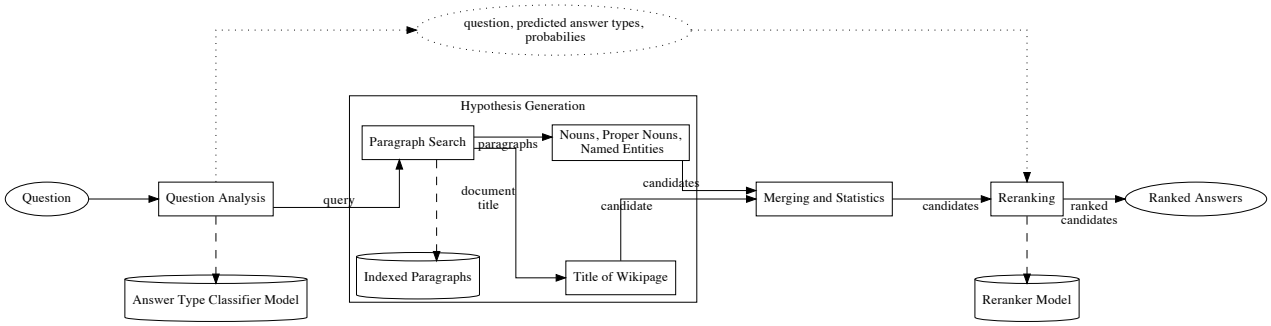

Figure 1: System overview.

indexed them using *Lucene* (Apache Lucene 2013). We applied Lucene's *Swedish Analyser* to stem the words and remove the stop words, both when indexing and when querying.

Lucene combines a Boolean model of information retrieval with the vector space model. By default, Lucene's vector space model uses TF-IDF weights. The vector space model score of document d, here a paragraph, for query q, here a question, is the *cosine similarity* of the TF-IDF weighted query vectors $V(q)$ and $V(d)$. Lucene then reweights the cosine similarity with its *Conceptual scoring formula*.

Given a question, the hypothesis generation module retrieves the 60 most relevant paragraphs using Lucene's scores. We set the upper bound to 60 to keep the system responsive enough with a near real-time performance on an ordinary laptop. For a complementary study on passage retrieval using the same corpus, see Pyykkö, Weegar, and Nugues (2014).

The module then applies the *Stagger* tagger (Östling 2013) to the text of the retrieved paragraphs to annotate the words with their parts of speech and named entities and parses it with a dependency parser (Nivre, Hall, and Nilsson 2006). Table 3 shows the entity types recognized by the tagger.

The hypothesis generation module extracts all the common nouns (NN), proper nouns (PM), and named entities from the retrieved paragraphs to build a list of candidates. A named entity can consist of multiple words while the other candidates can only be single words. We complemented each candidate object with additional information that we derived from the paragraphs. It includes the count, relative frequency, named entity type, the sentence the candidate occurred in with its part-of-speech tags and dependency graph, the Lucene score of the paragraph, Wikipedia article title, and paragraph text (Figure 2).

Table 3: The entity types recognized by the *Stagger* tagger.

person	work	product	inst	other
place	animal	myth	event	

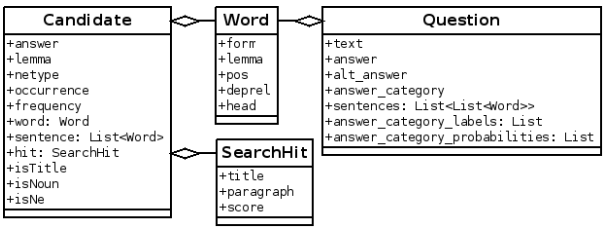

Figure 2: Candidate and question UML.

Most questions generate a considerable number of duplicate answer candidates. We considered two candidates to

be equal when their case-normalized lemmas were identical. We merged the corresponding objects using independent procedures for each object field: we added the counts; we kept the most relevant named entity type according to the Lucene score; we applied an *or* operator on the Boolean features. As a result, merging significantly reduced the number of candidates. Finally, the hypothesis generation returns a list of candidate objects.

As a consequence of the paragraph search described above, a significant number of words from the question often have both a high frequency in the most relevant paragraphs and a high Lucene score. Usually, those answer candidates are incorrect. A simple stop-word filter removing them proved effective.

Reranking

We first ranked the candidate answers by their counts in the 60 most relevant paragraphs. Although extremely simple, this proved surprisingly effective and we adopted it as a baseline.

We then applied a reranker to this list that we trained on the question corpus (Ravichandran, Hovy, and Och 2003). We submitted the questions through the candidate generation module and, given a question, we marked a candidate as positive if it corresponded to the correct answer, or negative if not. We trained a binary classifier using logistic regression and the LIBLINEAR library (Fan et al. 2008). Due to the unbalanced nature of the data, we only used questions having a correct answer in the candidates. Furthermore, we applied a uniform sampling to restrict the number of negative instances. In the end, the training data files consisted of about 500,000 instances, each having 29 features.

In addition to the nominal, numerical, and binary features shown in Figure 2, we used a bag of words of lexical features to build a contextual model. We extracted these lexical features from both the question context and the candidate answer context. Similarly to Comas, Turmo, and Màrquez (2010), we parsed the questions and the candidate contexts using a dependency parser. Figure 3 shows an example of a dependency graph for a corpus question. We extracted triples consisting of subject-verb-object (SVO) frame projections from the dependency graphs of the last sentence of the question text and from the paragraph sentences containing the candidate answers.

We added the first word of the question sentence to the bag of words as, usually, it is an interrogative determiner, pronoun, or adverb: *vem* 'who', *var* 'where', *när* 'when', etc. that is a hint at the answer type. For the lexical features, we built vocabularies for both the question and candidate contexts, where we kept the 50 most frequent words for respectively, the first word of the context, the subject, the verb, and the object. We represented the words not present in this vocabulary with a rare word flag. We also used a specific value to denote missing syntactic functions, for example the absence of a subject or an object in the sentence. Table 4 shows an overview of the features.

We finally used the probability outputs from the model as well as the Lucene score to rerank the baseline scores.

Results and Evaluation

We used a **correctness criterion** for the answers that we defined as a case-insensitive equality between the word form or the lemma in the candidate answer and the answer or the alternative answer provided in the question corpus. If one of the answer candidates generated for a question fulfills the correctness criterion, we consider the question as **answerable**.

In order to evaluate the candidate generation and the "answerability", we computed a histogram of the rank of the first paragraph that contains the answer to a question for all the questions in the corpus and the cumulative distribution of this histogram. This histogram is shown in Figure 4, where the y axes are normalized by the total number of questions. We can see that approximately 71% of the questions are answerable when we retrieve 500 paragraphs and considering all the questions. When considering only the single-word answers, up to 80% of the questions are answerable.

As we restricted the number of paragraphs in the hypothesis generation to the 60 most relevant ones, we can at most expect that 57% of all questions are answerable, and roughly 65% of the single word answers.

To evaluate the reranker, we looked at the rank distributions of the correct answers (Figure 5). The candidate ranks were computed using a 5-fold cross validation. The answer type prediction model was not part of the cross validation, i.e. it was trained using all the corpus questions. The upper plot shows the rank distribution of the baseline, and the lower plot, the rank distribution of the reranker. We can observe that the distribution shifted to the left in the lower plot. Both the median and the mean improved when using the reranker and we could reach a median 10 for the reranker down from 21 for the baseline. The number of questions with top ranked correct answers increased from the baseline's 155 questions to 224 questions for the reranker.

Conclusion

In this paper, we showed that we could build a question answering system using relatively simple components that could fit a language like Swedish for which less resources are available. We showed that a reranker could improve significantly the results, dividing the median by nearly two.

The *lexical answer type* is an indicator of the correctness of a candidate answer. This information was available for the questions and, because of the part-of-speech tagger we used, only for the named entities on the candidate side. Using a fine-grained answer type categorization might improve the results. Especially, if used together with a better type resolution for the candidate answer side. In addition, some questions with a fixed structure, such as *Vilken författare skrev... ?* 'Which author wrote... ?' have always the same answer type, human in this case. Such questions could be handled by specialized rules.

A way to improve the system performance would be to incorporate more knowledge sources, for example in location questions, and use external resources such as Geonames, DBpedia (Auer et al. 2007; Bizer et al. 2009), Yago (Suchanek, Kasneci, and Weikum 2007), or Freebase (Bol-

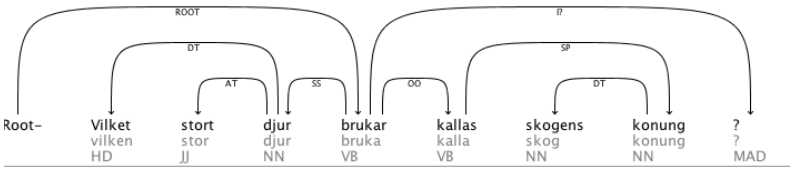

Figure 3: An example dependency graph for a question from the corpus: *Vilket stort djur brukar kallas skogens konung?* 'Which large animal is commonly called the king of the forest?' The graph shows both dependency relations, lemma, and part-of-speech tags. In this case, the SVO triple is *djur brukar kallas*.

Table 4: An overview the reranker features.

Type	Question	Candidate	Both
Nominal	Two best answer types	named entity type	–
Numerical	Two best probabilities	count, frequency, Lucene score, # words in answer	-
Boolean	isMultiChoiceQuestion	isTitle, isNoun, isNE, isNumeric, isDate	isWordInQuestion, isNEMatchingLAT (persons and places)
Lexical	first, SVO	first, SVO	Boolean comparisons

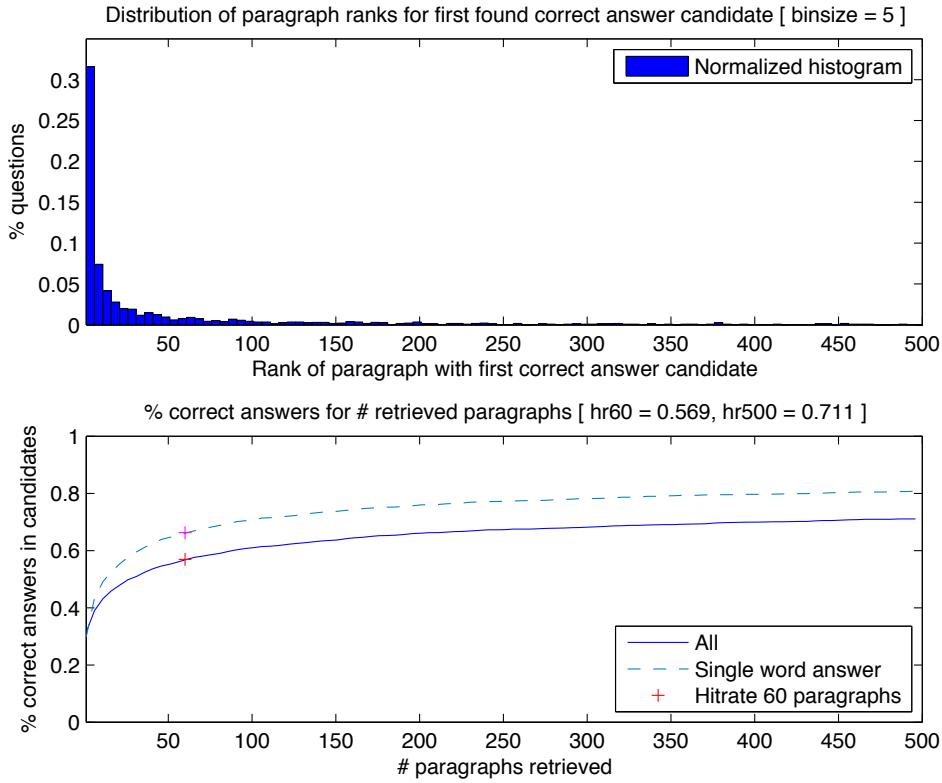

Figure 4: (*upper part*) Distribution of paragraph ranks for first found correct answer candidate. (*lower part*) Cumulative Distribution.

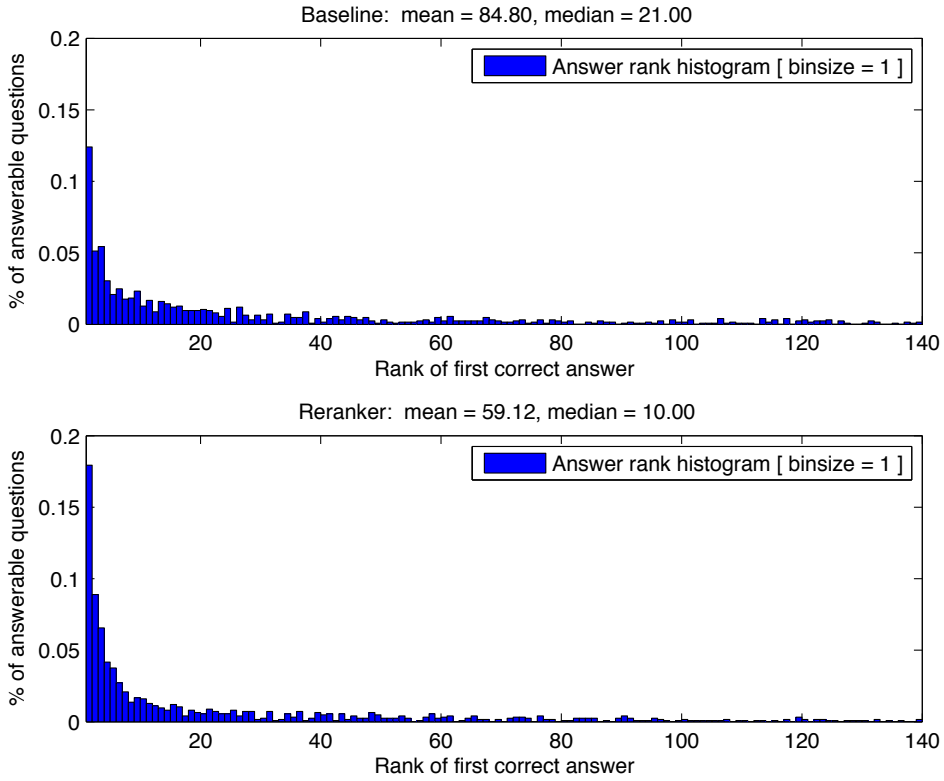

Figure 5: Distribution of candidate ranks for first correct answer candidate. Baseline (*upper part*) and Reranker (*lower part*).

lacker et al. 2008). Nonetheless, such databases were designed primarily for English and would require an adaptation to Swedish. Another option would be to use a cross-lingual question-answering framework (Ko et al. 2010).

As mentioned in Sect. *Previous Work*, more search strategies could be used: document search, custom similarity measures such as the distance between the question words and the answer candidate in the retrieved paragraph text, information measures such as pointwise mutual information, and plain text passage search. We could also use anchor text, the title from the Wikipedia articles as answer candidates, structured databases with SVO triples, and other relations to generate candidates.

For questions such as *Anja Pärson är uppvuxen i samma lilla fjällby som bl a Ingemar Stenmark. Vad heter den?* 'Anja Pärson grew up in the same small village than Ingemar Stenmark. What is it called?', *coreference resolution* could yield an improved performance by both adding a richer context to the question sentence and helping find the lexical answer type.

Using *better context models* for question text and for the sentences containing the candidates could also improve the performance. One approach to a more robust context modeling is to use word clusters instead of lexical features (Pinchak and Lin 2006), i.e. create a vector quantization of word-based contextual features including lexical, part-of-speech, and dependency information. A word cluster would then represent words often appearing in similar contexts.

Finally, we could add features to validate the answer candidate, e.g. the strength of an association between the candidate answer and key parts of the question such as the total number of search hits for a candidate answer and key parts of the question query using conventional web search.

Acknowledgments

This research was supported by Vetenskapsrådet, the Swedish research council, under grant 621-2010-4800, and the *Det digitaliserade samhället* and eSSENCE programs.

References

2013. Apache Lucene Core. http://lucene.apache.org/core/.

Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; and Ives, Z. 2007. DBpedia: A nucleus for a web of open data. In *The Semantic Web, Proceedings of 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007*, 722–735. Busan, Korea: Springer.

Bizer, C.; Lehmann, J.; Kobilarov, G.; Auer, S.; Becker, C.; Cyganiak, R.; and Hellmann, S. 2009. DBpedia—a crystallization point for the web of data. *Journal of Web Semantics* 154–165.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 1247–1250. ACM.

Chu-Carroll, J.; Fan, J.; Boguraev, B.; Carmel, D.; Sheinwald, D.; and Welty, C. 2012. Finding needles in the haystack: Search and candidate generation. *IBM Journal of Research and Development, Issue 3.4*.

Comas, P. R.; Turmo, J.; and Màrquez, L. 2010. Using dependency parsing and machine learning for factoid question answering on spoken documents. In *Proceedings of the 13th International Conference on Spoken Language Processing (INTERSPEECH 2010)*.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin., C.-J. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research* 9:1871–1874.

Fan, J.; Kalyanpur, A.; Gondek, D. C.; and Ferrucci, D. A. 2012. Automatic knowledge extraction from documents. *IBM Journal of Research and Development, Issue 3.4*.

Ferrucci, D. A. 2012. A framework for merging and ranking of answers in deepqa. *IBM Journal of Research and Development, Issue 3.4*.

Gondek, D.; Lally, A.; Kalyanpur, A.; Murdock, J.; Duboue, P.; Zhang, L.; Pan, Y.; Qiu, Z.; and Welty, C. 2012. A framework for merging and ranking of answers in deepqa. *IBM Journal of Research and Development, Issue 3.4*.

2013. jsoup: Java html parser. http://jsoup.org.

Ko, J.; Si, L.; Nyberg, E.; and Mitamura, T. 2010. Probabilistic models for answer-ranking in multilingual question-answering. *ACM Transactions on Information Systems* 28(3).

2013. Kvitt eller dubbelt – tiotusenkronorsfrågan. http://en.wikipedia.org/wiki/Kvitt_eller_dubbelt.

Li, X., and Roth, D. 2002. Learning question classifiers. In *COLING '02 Proceedings of the 19th international conference on Computational linguistics*, 1–7.

Nivre, J.; Hall, J.; and Nilsson, J. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC-2006*, 2216–2219.

Östling, R. 2013. Stagger: an open-source part of speech tagger for swedish. *Northern European Journal of Language Technology* 3:1–18.

Pinchak, C., and Lin, D. 2006. A probabilistic answer type model. *11th Conference of the European Chapter of the Association for Computational Linguistics* 393–400.

Pyykkö, J.; Weegar, R.; and Nugues, P. 2014. Passage retrieval in a question answering system. Submitted.

Ravichandran, D.; Hovy, E.; and Och, F. J. 2003. Statistical QA – classifier vs. re-ranker: What's the difference? In *Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering*, 69–75.

Suchanek, F. M.; Kasneci, G.; and Weikum, G. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, 697–706. ACM.

2008. Swedish wikipedia static html dump. http://dumps.wikimedia.org/other/static_html_dumps/current/sv/wikipedia-sv-html.tar.7z.

Thorsvad, K., and Thorsvad, H. 2005. Kvitt eller dubbelt.

Yu, H.-F.; Ho, C.-H.; Juan, Y.-C.; and Lin., C.-J. 2013. Libshorttext: A library for short-text classification and analysis. http://www.csie.ntu.edu.tw/~cjlin/papers/libshorttext.pdf.