Selectively Using Linguistic Resources throughout the Question Answering Pipeline

Raffaella Bernardi [a] Valentin Jijkoun [b] Gilad Mishne [b]
Maarten de Rijke [b]

[a] *KRDB Research Centre, Faculty of Computer Science, Free University of Bozen-Bolzano, Piazza Domenicani 3, 39100 Bolzano, Italy*

[b] *Language & Inference Technology Group, University of Amsterdam, Nieuwe Achtergracht 166, 1018 WV Amsterdam, The Netherlands*

Abstract

It is generally believed that question answering can benefit from natural language processing methods. So far, however, there have been few systematic studies of this conjecture. We report on ongoing work that is aimed at understanding the contribution of linguistically informed modules and resources to the overall performance of a generic question answering system. Specifically, we describe the ways in which currently we use linguistically motivated techniques, and demonstrate the impact of integrating, or not integrating, these techniques on the overall performance of our question answering system. Evaluation results are based on the TREC 2002 and TREC 2003 question sets.

Key words: Question Answering, Natural Language Processing, Evaluation

1 Introduction

For many years, researchers in information retrieval (IR) have tried to improve retrieval effectiveness through the use of natural language processing (NLP) techniques. By and large, the outcome of these attempts has been that most NLP methods do not improve IR effectiveness: NLP is often domain-specific and non-portable and effective IR depends upon properties of documents and queries that obviate some NLP techniques. Some have claimed that NLP errors hurt more than NLP techniques help [8], and that NLP is likely to be more useful for problems other than strict document retrieval [2]. *Question Answering* (QA) is believed to be such a domain.

In this paper we report on ongoing work aimed at gaining insights into the potential contribution of linguistically informed modules and resources to the question

answering process, and, in particular, to the performance of our QA system. We exploit natural language regularities at different levels. For instance, at the syntactic level, we use part-of-speech (POS) tagging of questions for identifying the question type. At the semantic level, we use hypernyms in WordNet both for question classification and off-line information extraction. Furthermore, for our participation in the TREC 2003 QA track we used POS information to transform list questions into factoid questions.

We provide three types of discussion on the usage of linguistic resources in our QA efforts. For some subtasks, such as classifying questions and pre-processing list questions, linguistic knowledge is absolutely essential to obtain reasonable performance, so we will not try to compare the end-to-end accuracy of the whole system with and without using NLP methods for such components; instead, we give a detailed description of the ways in which we use language resources in these components. A second type of discussion concerns modules (including off-line information extraction, table lookup, and answer pattern generation) where we do evaluate the performance of the QA system with different settings, to see the impact of using NLP techniques. And the third type concerns a comparison of the streams that together make up our QA system; each of these streams is a small QA system in itself, the main differences being the amount of linguistic information employed by the individual streams.

The remainder of this paper is organized as follows. We start by pointing to some related work, and describing our experimental set-up. Then, in Section 4 we describe modules for question classification and list question tranformations that cannot do without linguistic knowledge in our architecture. Then, in Section 5 we evaluate the impact of linguistic informedness on off-line answer extraction (and lookup) and search pattern generation. In Section 6 we compare the performance of different streams (embodying different QA strategies) for different question types. We conclude in Section 7.

2 Related Work

The belief that QA is an application in which sophisticated linguistic techniques will make a significant positive contribution has not undergone wide empirical verification so far. One system consistently manages to achieve high scores at the TREC QA task, using a complex architecture heavily employing NLP and AI technology [6]. But so far any attempts to replicate its results have failed. In general, though, linguistically-impoverished systems have outperformed those that attempt syntactic or semantic analysis. In support of those techniques, Katz and Lin [3] have looked closer at two linguistic phenomena in which syntactic relations do prove to be effective. In contrast, we don't restrict ourselves to questions potentially affected by specific linguistic phenomena, but eventually aim to understand the impact of

using linguistically informed techniques throughout the QA pipeline. In that sense, our paper is similar in spirit to [4] where the authors (amongst many other things) systematically explore the use of word overlap as a metric for ranking answers.

3 Experimental Setting

Most open domain QA systems implement some variation of a pipeline architecture with the following components: question analyzer (identifying the type of entity the question is looking for, along with the question focus), retrieval component (identifying documents that are likely to contain an answer), answer extraction (extracting entities of the appropriate type), and answer selection (filtering and reranking candidate answers, and selecting the best one).

Some answering strategies are beneficial to all question types, and others only for a subset. For this reason we implemented a *multi-stream* system: a system that includes a number of separate and independent subsystems, each of which is a complete standalone QA system that produces ranked answers, but not necessarily for all types of questions. The system's final answer is then taken from the combined pool of candidates. Our question answering system consists of several independent streams, ranging from linguistically informed to pattern-based to mostly data-driven:

Table Lookup: use pre-constructed specialized knowledge bases;
Pattern Match: search for answer patterns generated from a question (against the Web or against the TREC ACQUAINT corpus);
Tequesta: use Tequesta [7], a linguistically informed QA system for English; and
N-grams: generate multiple queries from the input question, using permutations of word n-grams, and retrieve snippets of relevant documents, either from the Web or from the corpus.

While each of the streams in our multi-stream architecture is a question answering system in its own right, they share a number of components, including a POS-tagger (we use Helmut Schmidt's TreeTagger), an NE-tagger, and filtering and reranking modules.

4 Using POS tags for Question Classification and List Questions

Question classification plays an important role in the performance of our QA system since it effects all subsequent steps. E.g., the type assigned to a question determines the table where we look for an answer, and also directs the pattern matching

and web answer streams, as well as Tequesta. Moreover, in the case of list questions, it also effects the question transformation rules, as we explain below. Therefore, it is essential to obtain optimal results for the question classification task and minimize the number of questions classified as `unknown`. Due to the limited complexity of natural language questions (compared to the full texts where the answers should be found), to tackle this problem we can exploit natural language syntax and semantics regularities. We use POS tagging and WordNet hypernyms, respectively.

Question Classification. The system associates an incoming question with one of the 31 question types. The question classification is done in two steps: first, using pattern matching rules and second, using POS tags and WordNet. Here we concentrate on the latter step, since this is where linguistic knowledge is at work. We identified a set of syntactic regularities which allowed us to extract the question's *target* and then search in WordNet for the corresponding type. For instance, using rules based on POS sequences we extract *company* from the question *1600. What automobile company makes the Spider?*, look for WordNet hypernyms of *company* matching one of our predefined types, and then classify the question as `organization`. As a case study we consider the performance of the POS-WordNet combination on a subset of the TREC 2002 questions. In total, 64 questions out of 500 were typed as `unknown` by the pattern matching strategy. Most of these questions were *what/which*-questions. All 64 questions received a type in the second classification step, and furthermore, 52 questions (81%) were typed correctly. The system failed to classify questions like *1572. For whom was the state of Pennsylvania named?* and *1633. What roller coaster is the fastest in the world?*, since it detected "state" and "coaster" as target words and assigned the incorrect types `organization` and `person`, respectively.

In general, pattern matching by itself is quite accurate in assigning fine-grained question-types corresponding to some fixed patterns. For instance, it correctly classified questions of types "also-known-as", "known-for", "date-of-birth", "date-of-death", "abbreviation". On the other hand, the combination of syntactic and semantic information performed well with respect to more general types such as "capital", "language", "organization", "person" and similar.

List questions. As of TREC 2003, list questions are part of the main task. Answering such questions is more difficult than answering general factoids, because the number of items to be returned is unknown and the system does not know when to stop searching. As a first step towards answering list questions, we reduce the problem to the problem of answering factoids, since a correct answer to a list question can be thought of as the list of the answers to a single "equivalent" factoid question. For instance, question *2034. List female astronauts or cosmonauts* can be answered by combining the answers given to the question *Who is a female astronaut or cosmonaut?* To this end, we use POS tagging to transform list questions into

factoid. A preliminary type is assigned to the list-questions by means of the POS-WordNet strategy explained above. Then, different transformation rules are applied to different types. For instance, questions typed as `person` are transformed into *who*-questions, whereas the ones typed as `organization` are transformed into *which*-questions. Furthermore, morphological transformation rules are applied to preserve number agreement. The factoid questions generated in this way are then treated as usual.

Summing up, linguistic knowledge proved to be of help in determining the focus words and the type of the questions. However, as for the latter task, it is not clear yet whether it can completely replace the more *ad-hoc* but safer pattern matching. In particular, it seems that the latter strategy performs better with respect to fine-grained question types, such as "date-of-birth" and "abbreviations." And in general, these two approaches to the question classification task appear to be complementary.

5 To Become Linguistically Informed or Not

We now describe some modules that can operate with or without being linguistically informed: off-line answer extraction, table lookup, and answer pattern generation. We evaluate the end-to-end performance of the system using these modules either with or without activating linguistic resources. By comparing the overall accuracy in these two settings, we aim to identify components that benefit from linguistically informed techniques.

Off-line Answer Extraction. One of the streams of our multi-stream QA system (*Table Lookup*) uses tables containing specific information extracted from the collection. Below, we describe the table lookup mechanism in detail. First, though, we focus on the impact of using linguistic knowledge during the *creation* of the tables. As a case study we consider the performance of the *Table Lookup* stream on the subset of the TREC 2002 questions asking about persons (e.g., *1440. Who was the lead singer for the Commodores?*, *1424. Who won the Oscar for best actor in 1970?* or *1565. What is Karl Malone's nickname?*). In total, 95 questions out of 500 are identified by the system (sometimes erroneously) as possibly referring to persons. For these questions the system tries to find answers in one of two relevant tables: *Roles* and *Leaders*. These tables have been generated automatically (and off-line) from the TREC collection by looking at some specific surface patterns around person names (as identified by the Named Entity Recognition module), such as apposition (*. . . Yusupha Ceesay, the manager of Elephant Walk, . . .*). While the *Leaders* table is populated solely based on part-of-speech and named entity information and on a small hand-built set of phrases identifying leadership (e.g., *head of state*, *queen*, *premier*), during the creation of the *Roles* table we have used WordNet to obtain a

fairly long list (2590 entries) of possible references to professions or occupations. The primary purpose of this list is as a filter to remove noise that might cause wrong answers.

Sophisticated machine learning methods [1] have recently been proposed for cleaning up the results of off-line information extraction in the QA setting. The question we are addressing, however, is whether *lexically informed filtering* can improve the performance of the whole QA system. We evaluated the performance of our *Table Lookup* stream in two ways: with WordNet-based filtering and without. Table 1 shows the evaluation results for the relevant 95 TREC 2002 questions.

Table 1
Evaluation of filtering for *Roles* table on 95 'person' questions.

Filtering	*Facts in the table*	*Total answers*	*Correct answers*
On	396558	41 (43%)	16 (17%)
Off	1614309	49 (52%)	17 (18%)

Not surprisingly, running the system on a version of the *Roles* table without WordNet filtering increases the number of answers given by the *Table Lookup* stream. While the overall accuracy of the *Table Lookup* stream (i.e., the number of correct answers divided by the number of questions) decreases from 39% for the filtered table to 35% for the non-filtered table, turning the filtering off allows us to answer one question more.

The reason why WordNet-based table clean-up does not help to improve (and even hurts) performance of the whole system, is that we use a separate module for answer filtering and selection *after* all streams have provided their answer candidates. We use both statistical and content-based methods to filter our strings that are not likely to contain the answer and to re-rank the remaining answer candidates, in a way similar to [5]. This post-filtering makes the system quite robust to noise among answer candidates. Hence, in the presence of such post-filtering methods, it is better to do off-line extraction in a lexically uninformed way: the increased recall helps more than the decreased precision hurts.

Table Lookup. As mentioned before, the format of the knowledge bases we extracted from the documents was simple text, where every line contains a few columns of unstructured text. We now describe the lookup process in these tables. We also indicate how lexical knowledge was used to improve the performance of this lookup.

When a question is classified as possibly having an answer in a table, we first identify the question keywords that will be used in the table search. Once the keywords have been identified, a line matching all of the words in the order they appeared

in the question is searched; if no line matches, we look again for a line containing all words, this time in any word order. If there is still no match, we start removing words from the list of words to match; the order of removal is based on the frequency of words in the language (i.e., common words are removed first) and POS tags (e.g. superlatives like *fastest, largest* are removed last). We do this until some threshold is reached (percentage of lookup words out of total keywords in the question). When a matching line is found, the text in the column that is declared to contain the information required as the answer is returned.

To identify the question keywords, we remove all stopwords from the question, using a standard English stopword list. Certain words should be treated differently in different table contexts, however: some words should be removed when looking up in a specific table but not from others, and so on. We therefore developed table specific hand-crafted lists of stopwords and *keepwords* — words which should not be removed when looking up. For example, for the Leaders table, words such as *vice* and *former* were considered keepwords, since removing them would alter the lookup meaning completely, while words such as *residents* or *population* were considered stopwords for the Inhabitants table, because we do not expect to find them in the table. Superlatives and numerals, such as *largest* and *third* were found to be keepwords for all tables. Table 2 presents results for the Table Lookup stream for a subset of 347 questions out of the 500 questions from TREC 2002; these questions were identified by the system as possibly having an answer in one of the tables.

Table 2
Evaluation of the Table Lookup with and without lexical knowledge

Stopwords, Keepwords	*Correct answers*	*MRR*
Without	97 (28%)	0.24
With	103 (30%)	0.25

Even with the relatively small manually created lists of stopwords and keepwords, using this heuristics allowed the system to answer 6 questions more. In our ongoing work we aim to generate table-specific stopwords and keepwords automatically, using a frequency analysis of words in the tables and comparing them to standard distributions of words in the language. Morever, we plan to use synonyms from WordNet to expand the original words from the question.

Pattern Generation The *Pattern Match* stream of our QA system uses words from the question to formulate regular expressions that could match possible answers in the text. For example, for the question *1400. When was the telegraph invented?* one possible pattern would be *the telegraph invented in ANSWER*, where *ANSWER* denotes the actual place the answer should be taken from. Our system uses a manually built set of rules for transforming questions into possible answer

patterns. To evaluate the applicability of a linguistically informed transformation procedure, we used two different sets of pattern formulation rules: one based only on the question type and actual question words, and another also using POS tags of the words of the question and a database of morphological variants of words. Table 3 shows the patterns generated using both sets of transformation rules for question *1419. What year did Alaska become a state?*.

Table 3
Patterns for question *1419. What year did Alaska become a state?*

Question type, words
`Alaska become a state ANSWER`
`Alaska become a state (in|on) ANSWER/`
Question type, words, part-of-speech tags, word forms
`Alaska becomes a state (in|on) ANSWER`
`Alaska becomes a state ANSWER`
`Alaska became a state (in|on) ANSWER`
`Alaska became a state ANSWER`

Since our pattern generation rules use only "shallow" information about the question (POS tags), in cases when syntactic structure of the question is complicated the resulting patterns may be ungrammatical. However, this does not add any noise to the system because these ill-formed answer patterns simply do not match and thus do not lead to incorrect answers.

We compared the performance of the *Pattern Match* stream with the two different set of question transformation rules on 500 TREC 2002 questions. The results are summarized in Table 4:

Table 4
Pattern Match stream with different pattern generation rules

Use PoS, wordforms	*Total answers*	*Correct answers*
No	30	4
Yes	39	13

As expected, using linguistically informed pattern generation method increases both the number of answer candidates and the number of the correct answers found by the pattern stream. At the same time, the number of incorrect answers remains the same, which means that this approach does not cause additional noise.

6 Comparison of Streams

In the previous section we took a close look at the use of linguistic resources at various specific stages in the QA process. In this section, we adopt a more global perspective and compare the results of some of the separate streams of our QA system. That is, to see the difference between linguistically-informed vs. redundancy-based approaches, we looked at the performance (the number of correctly answered questions) of different streams for different question types. The results of the comparison for four streams and several question types are shown in Table 5.

Table 5
Performance of different streams on different question types

Questions		Correct Answers			
Type	*# questions*	*Web N-grams*	*Web Patterns*	*Table Lookup*	*Tequesta*
date	82	21 (26%)	15 (18%)	20 (24%)	22 (27%)
location	101	21 (21%)	14 (14%)	7 (7%)	19 (19%)
agent	35	10 (29%)	2 (6%)	4 (12%)	3 (9%)
object	24	2 (8%)	1 (4%)	0 (0%)	0 (0%)
pers-ident	54	19 (35%)	5 (9%)	5 (9%)	7 (13%)
thing-ident	59	9 (15%)	2 (3%)	0 (0%)	0 (0%)

Whereas the *Web N-grams* stream relies entirely on data redundancy on the Web, the other three streams in the Table 5 make use of linguistic information, in one way or another (POS tagging, Named Entity Recognition, Wordnet, dependency parsing).

Interestingly, for some question types (*date, location* in Table 5) different streams perform quite similarly, and there are no clear "outsiders." For other question types (*agent, thing-ident*) Web-based aproaches easily outperform deeper methods.

It seems that linguistic processing can help for those questions where both the structure of the question (e.g., focus words) and the type of the answer (e.g., *location, person*) are clear and easily identifiable. On the other hand, for more syntactically and/or semantically "vague" questions like *What is the Stanley Cup made of?* (classified as *thing-ident*) or *Who makes viagra?* (classified as *agent*), the system cannot extract enough information to perform a meaningful linguistic analysis, and zero-knowledge statistical methods prove more useful. This suggests that linguistic and redundancy-based methods are to a large extent complementary, and that, ideally, the two approaches should be deployed in parallel.

7 Conclusions

Modern QA systems have a complex architecture and usually consist of a large number of inter-connected components. Complex dependencies between components make it very difficult to predict how a change in one module will affect the performance of the system as a whole. In this paper we studied the impact of bringing more linguistic knowledge into various components of our QA engine.

Our evaluations show that for some modules (question classification, answer pattern generation) using linguistically informed methods helps to significantly improve the overall accuracy, but making other components language-aware does not change the performance of the system or may be even harmful. Moreover, when examining the accuracy of the QA streams of our multi-stream architecture, we found that for different classes of questions either linguistically informed or redundancy-based methods can give better results, and for some question classes they perform at about the same level.

This suggests that a good system should cleverly combine both approaches. A careful study of the impact of each component on the end-to-end performance of the whole system (sometimes on specific subsets of input questions) is essential in order to choose the best architecture for different components in an informed way.

Acknowledgements

We would like to thank Stefan Schlobach for help and advice. This research was supported by the Netherlands Organization for Scientific Research (NWO) under project number 220-80-001. Maarten de Rijke was also supported by NWO under project numbers 612-13-001, 365-20-005, 612.069.006, 612.000.106, 612.000.207, and 612.066.302.

References

[1] M. Fleischman, E. Hovy, and A. Echihabi. Offline strategies for online question answering: Answering questions before they are asked. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 1–7, 2003.

[2] K. Sparck Jones. What is the role of NLP in text retrieval? In T. Strzalkowski, editor, *Natural Language Information Retrieval*. Kluwer, 1999.

[3] B. Katz and J. Lin. Selectively using relations to improve precision in question answering. In *Proceedings EACL 2003 Workshop on NLP for QA*, 2003.

[4] M. Light, G. Mann, E. Riloff, and E. Breck. Analyses for elucidating current question answering technology. *Natural Language Engineering*, 7:325–342, 2001.

[5] B. Magnini, M. Negri, R. Prevete, and H. Tanev. Comparing statistical and content-based techniques for answer validation on the web. In *Proceedings of the VIII Convegno AI*IA, Siena - Italy*, 2002.

[6] D. Moldovan, S. Harabagiu, R. Girju, P. Morarescu, A. Novischi, F. Lacatusu, A. Badulescu, and O. Bolohan. LCC tools for question answering. In *Proceedings TREC 2002*, 2003.

[7] C. Monz and M. de Rijke. Tequesta: The University of Amsterdam's Textual Question-Answering System. In *Proceedings TREC-10*, 2002.

[8] E. Voorhees. Natural language processing and information retrieval. In M.T. Pazienza, editor, *Information Extraction: Towards Scalable, Adaptable Systems*, volume 1714 of *LNCS*. Springer, 1999.