Type Checking in Open-Domain Question Answering

Stefan Schlobach and **Marius Olsthoorn** and **Maarten de Rijke**[1]

Abstract. Open domain question answering (QA) systems have to bridge the potential vocabulary mismatch between a question and its candidate answers. One can view this as a *recall* problem and address it accordingly. Recall oriented strategies to QA may generate considerable amounts of noise. To combat this, many open domain QA systems contain an explicit filtering or re-ranking component, which often check whether the answer is of the correct semantic type. Particular classes of questions expect specific answer types to which all of their answers should belong. We compare two kinds of strategies for answer type checking for open domain QA. One is redundancy-based and builds on the intuition that the amount of implicit knowledge which connects an answer to a question can be estimated by exploiting the redundancy of information available on the web. The other is knowledge-intensive, and exploits structured and semi-structured data sources to determine, with high confidence, the semantic type of suggested answers.

1 INTRODUCTION

Question answering (QA) is one of several recent attempts to realize information *pinpointing* as a refinement of the traditional document retrieval task. In response to a user's question, a QA system has to return an answer instead of a ranked list of relevant documents from which the user has to extract an answer herself. The way in which QA is currently evaluated at the Text REtrieval Conference (TREC, [22]) requires a high degree of precision on the systems' part [22]. Systems have to return *exact answers*: strings of one or more words, usually describing a named entity, that form the complete and non-redundant answer to a given question. This requirement gives QA a strong "high-precision" character. At the same time, however, open domain QA systems have to bridge the potential vocabulary mismatch between a question and its candidate answers. Because of this, *recall* is a serious issue for most QA systems. Now, the typical QA system architecture is a single processing stream that performs four steps in a sequential fashion: question analysis, search, extraction of answer candidates, and answer selection [8, 15, 17, 18]. The early steps are usually "non-exact" steps, aimed at maintaining recall at an acceptable level. The underlying assumption is that much of the noise picked up in the early steps will be filtered out in the final step. Thus, many systems contain a filtering or re-ranking component aimed at promoting correct answers and demoting incorrect ones.

We focus on one particular kind of filtering: answer type checking, that is, checking whether a given answer candidate belongs to the expected semantic type (or set of types). To many of the factoid questions used in the TREC QA track, a small number of semantic types can be associated that all reasonable answers to those questions can be expected to belong to. On top of the usually coarse-grained expected answer types used for extracting answer candidates (such as `PERSON`, `LOCATION`, `DATE` or `ORGANIZATION`), it is often possible (and necessary) to identify more precisely whether we are looking for, e.g., an `ACTOR`, a `CAPITAL`, a `YEAR`, or an `NGO` (Non-Governmental Organization). Today's named entity extraction technology is able to recognize the course-grained types with high accuracy, but recognizing fine-grained semantic types is not a solved problem. Moreover, many open domain QA systems work with noisy, incomplete, and often ungrammatical text snippets returned by search engines; on such input it is hard to obtain accurate recognition of semantic types. In this setting answer type checking can provide an important sanity check to weed out incorrect answers.

How can we operationalize answer type checking? We discuss two strategies, both using freely available on-line resources. First, for an increasing number of domains, such as geography, movies, and medicine, reliable wide-coverage structured or semi-structured data sources are available that can be used for answer type checking. Moreover, assuming these sources are available locally, access to them is efficient and answer type checking can be decomposed into shallow taxonomical reasoning steps combined with look-ups. For domains without such resources, or to make up for gaps in existing resources, redundancy-based approaches, bootstrapped with patterns capturing the expected answer types, can be used instead.

The aim of this paper is to study the potential impact of answer type checking on the performance of open domain QA systems. We break this general issue up into more manageable issues, each of which we aim to address in this paper: (1) Does type checking (generally) improve the performance of open domain QA systems? (2) How do knowledge-intensive and redundancy-base type checking compare? Are they complementary? Both questions have to be answered while keeping in mind that the external knowledge sources that we employ are usually domain-specific. Can our results be transferred from one knowledge source and domain to others?

We describe algorithms for knowledge-intensive and redundancy-based answer type checking (in Sections 3 and 4). To be able to assess knowledge-intensive answer type checking, we need access to fairly rich knowledge sources. For this reason, we carried out an experimental evaluation and comparison of the two methods on location questions. We report on our experiments in Section 5 and conclude in Section 6. We start out by discussing related work, in Section 2.

2 RELATED WORK

As with any information access task, recall oriented strategies to QA may generate considerable amounts of noise. To combat this, many systems participating in the TREC QA track contain an explicit filtering or re-ranking component, and in some cases this involves answer type checking. In their TREC 2002 system, BBN used a number of constraints to re-rank candidate answers [24]; one of these is checking whether the answer is of the correct location sub-type. LCC's QA system has an answer selection process that is very knowledge-intensive [16]. It incorporates lots of AI-like technology, by attempting to prove candidate answers from text, with feedback loops and sanity-checking, using extensive lexical resources. Other systems using knowledge-intensive type checking include IBM's (that uses the CYC knowledge base [2, 19]), and the University of Edinburgh's (that uses subtle reasoning mechanisms [4]). Some systems take the use of external knowledge sources a step further by relying almost

[1] Informatics Institute, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands. Email: {schlobac, olstrn, mdr}@science.uva.nl

exclusively on such sources for answers, and, as a final step, finding justification for the externally found answers in the text collection used as part of the TREC QA track [11]. While systems that find their answers externally use many of the same resources as systems that use knowledge-intensive answer type checking, they obviously use them in a different way, not as a filtering mechanism.

In contrast to these knowledge-intensive methods, we mention a redundancy-based approach due to Magnini et al. [12]. They employ the redundancy of the Web to re-rank (rather than filter out) candidate answers found in the collection, by using web search engine hit counts for question and answer terms. The idea is to quantitatively estimate the amount of implicit knowledge connecting an answer to a question by comparison of the number of co-occurrences of answer and keywords in the question. Our redundancy-based type-checking method refines this method by bootstrapping co-occurrence statistics with admissible answer types (Section 3 and 4).

Recently, several QA teams adopted complex architectures involving multiple streams that implement different answering strategies [2, 3, 5, 10, 9]. Here, one can exploit the idea that similar answers coming from different sources are more reliable, so the answer selection module should favor candidate answers found by multiple streams. In this paper we do not exploit this type of redundancy as a means of filtering or re-ranking, but refer to [5, 10, 1, 9].

3 ONTOLOGY-BASED TYPE CHECKING

Many QA-systems attempt to answer questions as follows: a *question type* is determined and relevant documents are retrieved from which a list of *candidate answers* is extracted. An *answer selection* process then orders the candidate answers and the top one is returned. The *expected answer type(s)* (or EAT(S)) of a question restrict(s) the admissible answers within a particular domain, such as geography, to more specific semantic classes, such as *river*, *country* or *tourist attraction*. These semantic classes are often organized in *ontologies*, which are, in the simplest case, just sets of concepts equipped with hierarchical relations. If we take the semantic concepts as our answer types, these ontologies provide the structure and the reasoning which is necessary to automate answer type checking.

If a candidate answer is known *not* to be an instance of any EAT associated with a question, it can immediately be excluded from the answer selection process. We will refer to this use of EATs as *answer type checking* by *filtering*. For filtering, a knowledge-intensive approach seems ideally suited: for each candidate answer we try to extract a *found answer type (FAT)* from knowledge and data sources, i.e., a most specific semantic type of which it is an instance. Because of the inherent incompleteness of knowledge and data sources in open domain applications, it may be impossible to determine a FAT for every candidate answer. Instead, we propose to determine the likelihood that the expected answer type is indeed a correct semantic type for a candidate answer and to *re-rank* the candidate answers according to this measure. For re-ranking, redundancy-based strategies are an obvious choice, the assumption being that the number of co-occurrences of answers and answer types allows us to quantify the relation between a question's EAT and a candidate answer.

Filtering. From various information sources we can often reliably determine a most specific concept of which a particular answer is an instance. In our case we use the GNS [7] and GNIS [6] databases and WORDNET [14] to relate geographical answers with one or more semantic types (its found answer type (FAT)). As answers may be ambiguous, we often need to associate a number of FATs to each candidate answer. Then, a candidate answer A is *correctly typed* for a question Q if one of the found answer types of A is compatible to one of the expected answer types of Q. The notion of compatibility need not be symmetric. Given the found and expected answer types and a notion of compatibility of two types F and E, the basic algorithm for filtering is as follows:

Extract the expected answer types of each question Q;
for each candidate answer A
  extract the found answer types for A;
  if there is an EAT E of Q and a FAT F of A, such that
    F and E are compatible
    then A is correctly typed;
return the correctly typed candidate answer in the original order;

Figure 1 shows an ontology with concepts *thing*, *city*, *state*, *capital* and *river*, where *capital* is more specific than *city*. Furthermore, let the question *Which is the biggest city in the world?* have the expected answer type *city*. Assume we find in an external data-source

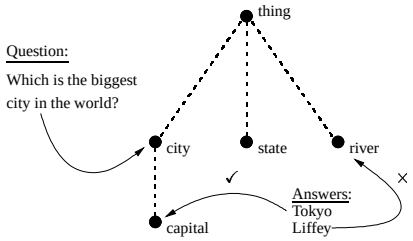

Figure 1. Filtering

that *Tokyo* is of type *capital*. To establish that *Tokyo* is a correctly typed answer we simply have to check that the type *capital* is compatible with, e.g., more specific as, type *city*. A different candidate answer *Liffey*, however, which is classified as a *river*, is incorrectly typed.

Re-ranking. Re-ranking does not depend on a found answer type of a candidate answer A but on the likelihood that A is both correct and has an answer type E. More precisely, any implementation has to define a measure for the correctness of the type E for A which needs to be merged with the confidence or the ranking of A (or both) determined in the answer selection process. Our measure of correct typing is defined by co-occurrence of a description of the EAT and the candidate answers on the web. The general strategy for *re-ranking* is then:

---

for each candidate answer A of a question Q
  for each expected answer type $E \in EAT(Q)$
    calcalculate the likelihood that A is
      correct and that E is the correct type,
      based on the
        probability that E is a correct type for A, and the
        original confidence and/or the rank of A
return the candidate answers reordered by this likelihood

Requirements. The strategies for type checking outlined above require some basic ingredients: an ontology of answer types has to be formalized, and each question has to be mapped to an expected answer type (EAT); furthermore, every implementation of filtering has to provide mechanisms to determine FATs and a notion of compatibility of FATs and EATs. Finally, an implementation of re-ranking must define measures for the likelihood of correct typing.

4 BUILDING A TYPE CHECKER

To assess the effectiveness of type checking we implemented filtering and re-ranking for the geography domain and added it to our own QA system, QUARTZ [20]. We take our (expected) answer types from the set of synsets in WORDNET [14]. Synsets are sets of words with the same intended meaning (in some context). They are hierarchically ordered by the hypernym (more general) and hyponym (more specific) relations. A *sibling* of a type T is a hyponym of a hypernym which is different from T. Finally, the *ancestor (descendant)* relation is the transitive closure of the hypernym relation (the hyponym relation, respectively). WORDNET contains many synsets in the geography domain. There are general concepts describing administrative units (e.g., cities or counties) or geological formations (e.g., volcanoes or beaches), but also instances such as particular states or countries. WORDNET provides an easy-to-use ontology of types for answer type checking in the geography domain. We will benefit two-fold: first, we can make use of the relative size of the geography fragment of WORDNET, and secondly, we get simple mappings from the questions and the candidate answers almost for free. Let us describe first how we map questions to expected answer types.

Expected Answer Type Extraction. We determine EATs of a question by a simple pattern-based analysis of the questions. Using WORDNET, we created a list l of about 300 geographical features describing elements of nature, structures, or administrative units. A simple strategy is to study the first feature f out of the list l which occurs in a question Q. In most cases the synsets containing f are reasonable EATs for Q. Note, that there could be more than one EAT for one feature f as we do not do any Word-Sense Disambiguation. Only few EATs are not found by this simple strategy. Take, for example, questions such as "Dublin is the capital of which country?", where the EAT is the synset containing the feature "country" (and not "capital"). Answer type extraction with hand-crafted patterns achieves a reasonable accuracy for most of our location questions (>90%). However, more robust EAT extraction methods need to be investigated to make the approach portable to other domains.

Knowledge-Intensive Filtering. Our implementation of knowledge-intensive filtering uses WORDNET and two publicly available "Name Information Systems." The Geographic Names Information System (GNIS) contains information "about almost 2 million physical and cultural features throughout the United States and its Territories" [6]. Sorted by state, GNIS contains information about geographic names, including the county, a feature type, geographical location etc. The GEOnet Names Server (GNS) "is the official repository of foreign place-name decisions ... and contains 3.95 million features with 5.42 million names" [7].

Some candidate answers, such as "Germany" to TREC question *1496, What country is Berlin in?* occur directly in WORDNET. Given our use of WORDNET as ontology of types we choose all the synsets containing the word "Germany" as found answer types (FATs). Unfortunately, this case is an exception: most names are not contained in WORDNET. The GNS and GNIS databases, however, associate almost every possible geographical name with a geographic feature type. Both GNIS and GNS provide mappings from feature types to natural language descriptions. The FATs determined by the data-base entries are therefore simply those synsets containing precisely these descriptions. In very few cases we had to adapt the coding by hand to ensure getting the intended WORDNET synsets. A simple example is the GNS type PPL, referring to a *populated place*, which we map to the synsets containing the word "city". In the case of complex candidate answers, i.e., answers containing more than one word, the FATs are separately determined for the complex answer string as well as for its constituent given the methods we described above.

We also need a notion of compatibility between answer types. Our definition is based on the WORDNET hierarchy: a FAT F is *compatible* with an EAT E, abbreviated by $comp(F, E)$ if it is more specific than or equal to the EAT.[2] Formally, this means that $comp(F, E)$ holds if, and only, if E is an ancestor of F, or if E equals F.

Redundancy-Based Re-ranking. No data source can be complete; in the previous section we presented a strategy for re-ranking based on the likelihood that the EAT of the question is the semantic type for a candidate answer. We implemented a purely redundancy-based method that is based on the assumption that co-occurrences of words in a large corpus (such as the web) can provide significant statistical information about the relation between words. We define two simple measures for the correctness of an EAT with respect to a candidate answer A based on the probability that a description E of the EAT and A occur together in the same documents. Remember that $EAT(Q)$ is a set of synsets, which were chosen in the EAT extraction process, because the question Q contains a particular feature f. For our algorithm we simply choose this feature f as the canonical description E of the synset. The number of web-pages a term T occurs in will be called the *hit-count* of T and abbreviated as $hc(T)$. $MaxHits$ is the total number of web pages indexed by our search engine. The two measures are then defined as follows.

1. *Conditional type probability*: $CTP(E, A) = P(E|A) = \frac{hc(E+A)}{hc(A)}$ is the probability that the expected answer type E occurs in a document given that it contains the candidate answer A. The assumption is that the EAT is often used to describe the most important properties of an instance. CTP is very similar to Pointwise Mutual Information (PMI) [13]. As long as there is only one EAT for a question, it produces the same ranking because the hit-counts for the EAT remain constant.
2. *Normalized conditional type probability:* $NCTP(E, A) = \frac{P(E|A)}{P(E|\neg A)} = \frac{hc(E+A)(MAX\_HITS - hc(A))}{hc(A)(hc(E) - hc(E+A))}$. A drawback of conditional type probability is that it only works if (as we expect) the answer implies the type. It may happen that the answer itself occurs very often in a different context and does not always imply the answer type. E.g., this occurs if a particular string refers to several different answers. The normalized conditional type probability compensates for this.[3]

With CTP and normalized CTP we implemented two measures which are natural in the context of linking an expected answer type and a candidate answer. To re-order the set of candidate answers we also need strategies to combine the new measures with the confidence $conf(A)$ and ranking $rank(A)$ that the answer selection process gave to a candidate answer A. Let $meas(E, A)$ be one of the previous methods $CTP(E, A)$ or $NCTP(E, A)$. We implemented several strategies of which the following proved to be the most effective. *Equal merging* combines the measure for E and A and the confidence of A: $meas(E, A) * conf(A)$. And *Reciprocal merging* combines the reciprocal original rank with the new measure of confidence of the typing: $meas(E, A) * \frac{1}{rank(A)}$.

[2] The notion of compatibility can be more complex depending on the ontology, the reasoning and the available data sources. If we consider more complex answer types, such as conjunctive ones, we may have to relax the definition. E.g., assume we have a complex EAT *river & German & muddy*. In this case an answer with a FAT *German & river* could be compatible to the EAT even though the FAT is not more specific than the EAT.

[3] We implemented additional, more complex measures such as the ones introduced in [12], but none of them improved over NCTP in our experiments.

5 EXPERIMENTAL EVALUATION

One of the main goals of our type checker was a proof of concept, i.e., to evaluate whether type checking could actually be successfully integrated in a QA system, at least for a particular domain. However, because the specialization to a particular domain is very much against the spirit of open domain QA, we had to address the issue of incompleteness of knowledge sources by introducing both redundancy based and knowledge-intensive strategies for type checking. In this section we discuss our experience with filtering and re-ranking given a number of experiments we conducted. Particular focus was put on the following issues: What are the advantages of each of the two strategies? And: What helps and what hinders each method?

Experiments We evaluated the output of our QA system on 261 location questions from previous TREC QA topics and on 578 location questions from an on-line trivia collection [21], both without and with type-checking. These were fed to our own QUARTZ system, and the list of candidate answers returned by it was subjected to answer type checking. We used two evaluation measures: the Mean Reciprocal Rank (MRR: the mean over all questions of "1 over the rank of the first correct answer, if any") [23], and the number (and percentage) of correct answers. For 594 of the 839 questions we determined over 40 different expected answer types. The types *country*, *city*, *capital*, *state*, and *river* were the most common and assigned to over 60% of the questions. We evaluated the EAT extraction process by hand and found an accuracy of over 90%.

To establish a realistic upperbound on the performance of answer type checking, we went through all the questions and their candidate answers by hand and checked how much *human type-checking* improves the results. Then we compared the performance of knowledge intensive filtering and redundancy based re-ranking (RBRR). To study the influence of the use of databases on filtering, we ran both the algorithm described in the previous section and a version using only WORDNET to find the FATs. This latter method will be denoted by KIF-WN below, the full version simply as KIF. For re-ranking we combined the two measures *CTP* and *NCTP* with *equal* and *reciprocal merging* to get the 4 runs: RBRR-CTP-Equal, RBRR-CTP-Recip, RBRR-NCTP-Equal and RBRR-NCTP-Recip.

Results Table 1 summarizes the main empirical results of our experiments. For each of the strategies and implementations we give the total number and the average of correct answers, as well as the MRR. The results show that type checking is useful, and that it can successfully be applied. Knowledge intensive filtering can significantly improve the overall performance of a QA system for geography questions, but even the best available strategy performs significantly worse than a human expert. What surprised us was that adding the GNIS and GNS database to the filtering lead to a substantial drop in performance compared to the filtering method based on WORDNET alone. Finally, redundancy based re-ranking failed to make a significant positive difference on the overall performance, which even dropped for the reciprocal merging strategy.

Strategy	correct answers	% of correct answers	Average MRR
No type-checking	244	29%	0.33
Human type-checking	331 (+36%)	36.4%	n/a
KIF	271 (+11%)	32.3%	0.37
KIF-WN	292 (+20%)	34.8%	0.38
RBRR-CTP-Equal	248 (+2%)	30%	0.34
RBRR-CTP-Recip	229 (-6%)	27%	0.31
RBRR-NCTP-Equal	249 (+2%)	30%	0.34
RBRR-NCTP-Recip	230 (-6%)	27%	0.31

Table 1. Overview of experimental results.

Error Analysis To obtain a better understanding of our results we look at them in more detail.

In which cases does knowledge intensive filtering fail? Knowledge intensive filtering does not achieve the potential improvement of 36% obtained with human type-checking, both because we do not exclude enough answers and because we incorrectly exclude correct answers. There are three conceptually interesting reasons for these errors. First, *incompleteness*: there are rare question types, such as *tourist attraction* or *motorway*, where the candidate answer (and its type) is simply unknown to the knowledge sources. In other cases insufficient information is available about a specific instance. E.g., neither WORDNET, nor GNS and GNIS, contain the information that "Rotterdam" is a port, or "Palermo" a regional capital. Second, *representation*: some errors are directly due to the way information is represented in WORDNET. E.g., the synset {Quito}, which has a synset containing "country" as ancestor (via hypernyms *capital, seat* and *center*). This is due to the ambiguity of the term "country" which denotes both a political entity and a region. And third, *linguistics*: we sometimes fail to map answers to the correct synset. E.g., the answer "Indian" to the question *In which ocean are the group of islands called the Seychelles?* is not mapped to the correct synset {Indian Ocean}. Finally, the answer "African" to the question *What is the second largest continent in the world?* is mistakenly excluded because we do not map the adjective to the corresponding noun.

Where does it harm to use databases? A surprising result was that the inclusion of information from the GNS and the GNIS database decreased the accuracy of filtering. There are two main reasons. First, WORDNET *is sufficiently ignorant*: consider the question *What province is Montreal in?* where the candidate answer "Shanghai" was originally higher ranked that the correct answer "Quebec". The database GNIS knows that Shanghai is a province whereas WORDNET does not, and (incorrectly) excludes "Shanghai" as falsely typed. Second, *data is highly ambiguous*: knowledge intensive filtering with the GNIS and GNS database may fail because countries are usually not filtered from questions with EAT containing the word "city". The reason is that almost every country name, such as "France, China, Island" or "Scotland", is also a name of a city.

The examples show that filtering with WORDNET alone combines type-checking and sanity checking. This works fine because both our question set and WORDNET are biased toward questions about Europe or America. However, for more specific questions we will need more information as a fall-back because we cannot use the incompleteness WORDNET as a "sanity check" any more. In this case our primary task is to reduce the ambiguity of the data in the databases.

Why does re-ranking fail to make a positive impact? Re-ranking with our best re-ranking strategy RBRR-NCTP-Equal brings the correct answer to the top in 29 cases. But there are almost as many questions (25) for which the correct answer is mistakenly ranked lower. Here are two examples. First, *semantics versus word-usage:* for knowledge intensive filtering with databases, candidate answers are often semantically ambiguous. An example is the correct answer *M6* to the question *Which motorway links Birmingham and Lancaster?*. The term M6 has many readings, of which the one referring to the motorway is only a very special one (that simply

drowns amidst other readings). Second, *complex answers produce non-representative hit-counts*: the previous observation can be generalized. The more specific a candidate answer the more likely it is to co-occur with the expected answer type. This is because it gets semantically more constrained and the natural semantic ambiguity of the candidate answer is reduced. A number of errors of our re-ranking strategy can be traced back to this problem: whenever a name is constrained in any way, such as in *Eastern Afghanistan*, the likelihood that it is correctly typed increases. The effect of this problem is magnified in our experiments, because our QA system has a tendency to produce very specific candidate answers (which have low hit-counts).

Lessons learned There is little we can do about the incompleteness of the knowledge sources and the way the information is represented. To improve knowledge intensive filtering we will therefore have to focus on more linguistically informed methods to improve the quality of the mappings from questions to EATs, and from candidate answers to FATs. A more technical problem is to counterbalance the effect that complex answers produce non-representative hit-counts. A closer look at the experimental results show that the NCTP scores for the correct answer are usually better than for CTP. But although we introduced NCTP for this purpose the results are not good enough yet. There are two things we can do: to define a new measure which gives a stronger impetus to more common candidate answers, and to avoid the negative effect of the tiling to the type-checking. Conceptually more interesting is the problem of reducing semantic ambiguity.

Combining methods to reduce semantic ambiguity. Both knowledge intensive filtering with databases and redundancy-based re-ranking suffer from semantic ambiguity: candidate answers can be interpreted in many ways. Prospective solutions could use the question's EAT to provide context for restricting the possible interpretations of candidate answers. In both cases knowledge intensive and redundancy-based methods have to be combined: to disambiguate FATs for filtering we propose to use hit-counts to establish the confidence in the FATs, and disambiguation for redundancy-based type-checking makes use of sets of types that are 'near' the EAT in WORDNET. We omit the details due to lack of space.

6 CONCLUSION

Many open domain QA systems answer questions by first extracting a huge number of candidate answers from a document collection, and then picking the most promising one from the list. One criteria for this answer selection is whether the candidate answer is of the semantic type which is expected by the question. We presented two strategies for answer type checking, *filtering* and *re-ranking*, which we implemented using knowledge intensive methods for the former, and a redundancy-based approach for the latter. Our experimental findings clearly show the merits of answer type checking in general, but there is a mixed message about the two approaches. Knowledge intensive filtering substantially improves the number of correct answers, but redundancy-based re-ranking does not. Interestingly, the inclusion of two enormous data-sources actually leads to a decrease in performance. Both problems can be explained by the semantic ambiguity of the candidate answer, in both cases the types of candidate answers are determined incorrectly because no use is made of the context provided by the question. In current research we combined knowledge intensive and redundancy-based approaches, where an implementation of one of them showed first promising results.

Our evaluation is very specific for the large class of questions about geography that we considered — this is an ideal domain for knowledge intensive approaches. Nevertheless, we believe that our approach can be ported to other domains. As we made explicit in the paper, what is required is an ontology of types, mechanisms to extract the EATs and mappings from candidate answers to FATs. Fortunately, the redundancy-based approach is domain independent, so that we expect to be able to apply substantial parts of our general strategy in other domains.

ACKNOWLEDGEMENTS

We thank Gilad Mishne 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] J.-B. Berthelin et al., 'Getting reliable answers by exploiting results from several sources of information', in *Questions and Answers: Theoretical and Applied Perspectives*, eds., R. Bernardi and M. Moortgat, (2003).
[2] J. Chu-Carrol et al., 'A multi-strategy and multi-source approach to question answering', in *Proceedings of TREC 2002*, (2003).
[3] C.L.A. Clarke et al., 'Statistical selection of exact answers', in *Proceedings of TREC 2002*, (2003).
[4] T. Dalmas and B. Webber, 'Information fusion for answering factoid questions', in *Questions and Answers: Theoretical and Applied Perspectives*, eds., R. Bernardi and M. Moortgat, (2003).
[5] G. de Chalendar et al., 'The question answering system QALC at LIMSI: experiments in using Web and WordNet', in *Proceedings of TREC 2002*, (2003).
[6] Geographic Names Information System. `http://geonames.usgs.gov/stategaz/index.html`.
[7] GEOnet Names Server. `http://gnswww.nima.mil/geonames/GNS/index.jsp`.
[8] E. Hovy, H. Hermjakob, M. Junk, and C.-Y. Lin, 'Question answering in Webclopedia', in *Proceedings TREC-9*, (2000).
[9] V. Jijkoun and M. De Rijke, 'Answer selection in a multi-stream open domain question answering system', in *Proceedings ECIR 2004*, LNCS. Springer, (2004).
[10] V. Jijkoun et al., 'The University of Amsterdam at TREC 2003', in *TREC 2003 Notebook Papers*, (2003).
[11] J. Lin and B. Katz, 'Question answering from the web using knowledge annotation and knowledge mining techniques', in *Proceedings CIKM 2003*, (2003).
[12] B. Magnini et al., 'Is it the right answer? Exploiting web redundancy for answer validation', in *Proceedings ACL 2002*, pp. 425–432, Philadelphia, (2002).
[13] C. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*, MIT Press, Cambridge, MA, 1999.
[14] G.A. Miller, 'WordNet: A lexical database', *Communications of the ACM*, **38**(11), 39–41, (1995).
[15] D. Moldovan et al., 'The structure and performance of an open domain question answering system', in *Proceedings ACL 2000*, pp. 563–570, (2000).
[16] D. Moldovan et al., 'Performance issues and error analysis in an open-domain question answering system', *ACM Transactions on Information Systems*, **21**, 133–154, (2003).
[17] C. Monz and M. de Rijke, 'Tequesta: The University of Amsterdam's textual question answering system', in *Proceedings TREC 2001*, pp. 519–528, (2002).
[18] J. Prager et al., 'Question-answering by predicitive annotation', in *Proceedings SIGIR 2000*, pp. 184–191, (2000).
[19] J. Prager et al., 'IBM's PIQUANT in TREC 2003', in *TREC 2003 Conference Notebook*, pp. 36–45, (2003).
[20] QUARTZ. `http://ilps.science.uva.nl/~qa`.
[21] `http://www.quiz-zone.co.uk/`.
[22] Text REtrieval Conference (TREC). `http://trec.nist.gov`.
[23] E.M. Voorhees and D.K. Harman, 'Appendix: Common Evaluation Measures', in *Proceedings of TREC 2002*, (2003).
[24] J. Xu et al., 'TREC 2002 QA at BBN: Answer Selection and Confidence Estimation', in *Proceedings of TREC 2002*, (2003).