Scaling Answer Type Detection to Large Hierarchies

Kirk Roberts and Andrew Hickl

Language Computer Corporation
1701 North Collins Boulevard Suite 2000
Richardson, Texas USA
{kirk,andy}@languagecomputer.com

Abstract

This paper describes the creation of a state-of-the-art *answer type detection* system capable of recognizing more than 200 different expected answer types with greater than 85% precision and recall. After describing how we constructed a new, multi-tiered answer type hierarchy from the set of entity types recognized by Language Computer Corporation's CICEROLITE named entity recognition system, we demonstrate how we used this hierarchy to annotate a new corpus of more than 10,000 English factoid questions. We show how an answer type detection system trained on this corpus can be used to enhance the accuracy of a state-of-the-art question-answering system (Hickl et al., 2007; Hickl et al., 2006b) by more than 7% overall.

1. Introduction

Work in factoid question-answering (Q/A) has long leveraged *answer type detection* (ATD) systems in order to identify the semantic class of the entities, words, or phrases which are most likely to correspond to the exact answer to a natural language question. For example, given a question like *Who was responsible for coordinating disaster relief for victims of Hurricane Katrina?*, ATD components enable Q/A systems to retrieve the sets of candidate answers which include the INDIVIDUALS and/or ORGANIZATIONS who provided aid to the victims of the hurricane.

Early work in ATD (Harabagiu et al., 2000; Harabagiu et al., 2001) leveraged sets of heuristics in order to identify the expected answer types (EATs) of questions submitted to a Q/A system. Most modern Q/A systems, however, follow work done by (Li and Roth, 2002) in using machine learning classifiers in order to select the one (or more) EATs from a fixed hierarchy of answer types which are most appropriate for a particular question.

While learning-based approaches have dramatically increased the precision of open-domain ATD systems, most current ATD components have only been tasked with distinguishing amongst a limited set of EATs. For example, the most commonly-used answer type hierarchy (ATH), the University of Illinois (UIUC) answer type hierarchy created by (Li and Roth, 2002), includes only a total of 50 unique expected answer types (generally referred to as "fine" answer types), organized into 6 different categories (referred to as "coarse" answer types). (The entire UIUC ATH is presented in Table 1.)

While ATD systems based on the the UIUC hierarchy has been employed by a number of participants in recent TREC[1] Question-Answering evaluations, we believe that the size (and coverage) of current answer type hierarchies represents a factor which significantly limits both the performance of open-domain factoid question-answering systems and the number of questions that a Q/A system can be used to answer. Without coverage for a specific answer type within its ATH, a Q/A system must retrieve – and extract – answers using other answer types from its ATH which are conceptually "nearest" to the expected answer type of the question. For example, an ATH without a unique EAT for ORGANIZATIONS may be forced to search for answers using similar types such as PERSON or NATIONALITY. In a similar fashion, limited ATHs can also increase the challenge of extracting exact answers from retrieved text passages. For example, while most ATD systems can identify when a question is seeking a INDIVIDUAL as its EAT, a more articulated ATD system capable of distinguish between classes of individuals (e.g. CEO, BASEBALL-PLAYER, POLITICIAN, RELIGIOUS LEADER) could greatly reduce the total number of (spurious) candidate answers retrieved for the question. Systems without such an articulated ATH would be required to recognize valid answers from a much larger list of possible candidates.

Coarse Type	Fine Types
ABBREVIATION	ABBREVIATION, EXPANDEDABBREVIATION
ENTITY	ANIMAL, BODY, COLOR, CREATIVE, CURRENCY, DISEASEMEDICINE, EVENT, FOOD, INSTRUMENT, LANGUAGE, LETTER, OTHER, PLANT, PRODUCT, RELIGION, SPORT, SUBSTANCE, SYMBOL, TECHNIQUE, TERM, VEHICLE, WORD
DESCRIPTION	DEFINITION, DESCRIPTION, MANNER, REASON
HUMAN	GROUP, INDIVIDUAL, TITLE, DESCRIPTION
LOCATION	CITY, COUNTRY, MOUNTAIN, OTHER, STATE
NUMERIC	CODE, COUNT, DATE, DISTANCE, MONEY, ORDER, OTHER, PERIOD, PERCENT, SPEED, TEMPERATURE, SIZE, WEIGHT

Table 1: The UIUC answer type hierarchy.

In this paper, we address the challenge of creating – and learning – large hierarchies of answer types for open-domain question-answering applications. We describe the creation of a new, state-of-the-art ATD system capable of recognizing more than 200 different expected answer types with greater than 85% precision and recall. We first describe how we created a coherent ATH from the more than 350 different named entities recognized by Language Computer Corporation's (LCC) CICEROLITE named entity recognition system. We then describe how we used this large ATH in order to annotate a corpus of more than 10,000 English questions mined from web documents and show

[1] Text Retrieval Conference. For more information, see http://trec.nist.gov.

how the hierarchy we propose can boost the accuracy of a state-of-the-art question-answering system (Hickl et al., 2006a) by more than 7% overall.

The rest of this paper is organized in the following way. Section 2 presents a discussion of previous heuristic- and classification-based approaches to the problem of ATD for open-domain Q/A systems. Section 3 describes how we created a large, multi-tiered ATH from the sets of entity types recognized by LCC's CICEROLITE. Section 4 describes our efforts to use this hierarchy to annotate a large corpus of questions. Section 5 provides details of our ATD system. Section 6 details results of evaluations using our proposed ATH in conjunction with a state-of-the-art question-answering system, LCC's FERRET (Hickl et al., 2006a), and Section 7 presents our conclusions.

2. Previous Work

Answer type hierarchies were first employed by (Harabagiu et al., 2000) using a heuristic classifier based on the WORDNET (Miller, 1995) ontology. (Li and Roth, 2002) explored the use of machine learning techniques to answer type detection and (Krishnan et al., 2005) improved accuracy through the use of their informer span. Alternatively, (Pinchak and Lin, 2006) use a probabilistic model with no pre-defined hierarchy in order to identify the type of information sought by a factoid question.

We know of no previous work which combines the ability to scale to large ATHs and provide the benefits of a machine-learning based system. While the ATH in (Harabagiu et al., 2000) could easily be scaled to include a potentially very large number of types (e.g. see (Harabagiu et al., 2005) for an example of how this could be accomplished for a top-performing TREC Q/A system), it is constrained in its aapproach to WORDNET's hand-built hypernym relations, which does not coincide with common answer types in questions. in contrast, the (Li and Roth, 2002) UIUC ATH is designed especially for questions, but lacks the ability to extend the depth of the hierarchy when Q/A systems are capable of handling more detailed answer types. (See Section 3 for a more detailed discussion of the UIUC hierarchy.)

In order to create a new ATH capable of addressing the needs of today's open-domain factoid Q/A systems, we wanted to develop an ATH which was capable of scaling to any given set of named entity types. Our work in this paper sought to address the following requirements:

Requirement 1. The ability to add "abstract" answer types that represent some subset of other answer types. These would correspond to common question ambiguities, such as the above example of an unknown level of government or an ambiguity between individuals and groups such as *Who is in charge of the Hurricane Katrina relief effort?*.

Requirement 2. The ability to add new answer types as they arise anywhere in the hierarchy. This is a common case in an real-world environment where new question-answering requirements dictate new additions to the hierarchy.

Requirement 3. The ability to scale the hierarchy to any size without dramatically increasing the amount of training data or annotation complexity. The larger a hierarchy scales to, the less reasonable it is to expect an annotator to have a total understanding of the ATH.

3. Creating a Large Answer Type Hierarchy for Factoid Q/A

In this section, we describe how we created a large, multi-tiered hierarchy of over 200 expected answer types from the set of entity types recognized by LCC's CICEROLITE named entity recognition system.

We assume that the taxonomy of answer types included in an answer type hierarchy (ATH) can be represented as a tree structure of varying depth where branchings mark decision points between different semantic classes that all share a hypernym-like relationship with their parent. (A graphical representation of a portion of the LCC ATH is presented in Figure 1.)

The UIUC answer type hierarchy (Li and Roth, 2002) (and corresponding annotated corpus of nearly 6000 questions) has provided a solid test-bed for researchers to develop and benchmark a variety of machine learning techniques for answer type detection. While this corpus was designed to cover many of "common" question types, it only includes a total of 50 different answer types and does contain some notable gaps. For example, while the UIUC ATH DATE can be used in conjunction with many *when* questions, it does not provide the granularity necessary to answer more specific temporal questions, such as *What year was Clinton elected to the Senate?* or *What day of the week was Lincoln's birthday in 2002?*.

We feel that the time is right for work in ATD to move beyond the UIUC ATH and to begin to tackle problems of organizing and learning answer type hierarchies that encompass several hundreds of diverse expected answer types. We believe that this effort would be in line with recent work looking at a similar type of semantic categorization problem – named entity recognition – in which researchers have moved from using simple heuristics and classifiers to unsupervised or semi-supervised methods capable of inducing hundreds (if not thousands) of entity types from large collections of texts.

In our work, we used output from LCC's own, wide-coverage named entity recognition system, CICEROLITE in order to construct a novel ATH which included more than 200 different EATs. (A table listing some of entity types recognized by CICEROLITE is presented in Table 2.)

Purpose	Example(s)
ACE Types/Subtypes	PERSON/INDIVIDUAL, FACILITY/AIRPORT, GPE/COUNTRY, WEAPON/BLUNT
Common TREC Types	SPORTS-TEAM, WRITTEN-WORK DEATH-MANNER, DISEASE, ACRONYM
Temporal Reasoning	TIME, MONTH, YEAR-RANGE
Spatial Reasoning	MOUNTAIN, OCEAN, LATITUDE COMPOUND-LOCATION

Table 2: Examples from CICEROLITE's 350+ entity types

3.1. Entity Type Hierarchy vs. Answer Type Hierarchy

Differences between a hierarchy of extractable named entities (ETH) and the ATH are subtle but important. While

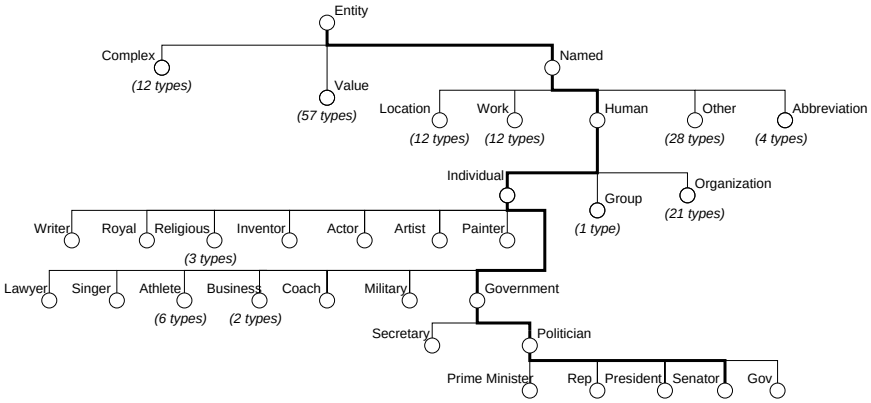

Figure 1: Sample hierarchy.

named entity types are the classes that may be *extracted*, answer types describe the user's *desired* class. This distinction can lead both to different hierarchy structure (such as the PERSON-ORG example in the previous section) as well as having nodes in the ATH that need not necessarily map directly to an entity type. For example, the question *How old is Condoleezza Rice?* clearly states the *desired* answer type requirement AGE. But the *extracted* types may additionally include NUMBER (*He turned 22 today.*) and DATE (*He was born on 5/5/89.*).

Furthermore, answers need not necessarily be drawn from entities. While some Q/A Systems detect the difference between "factoid" and "complex" questions (Hickl et al., 2006a), it is often desirable to include both in the ATH such as the DESCRIPTION coarse type in the UIUC hierarchy.

3.2. Constructing an ATH from an ETH

The following steps demonstrate how one could construct an answer type hierarchy from a pre-existing entity type hierarchy.

Step 1: Initialize. Create the initial ATH as a direct clone of the existing ETH.

Step 2: Consolidate. For every parent node in the hierarchy, if questions can ambiguously refer to a subset of the children, then combine that subset under a new node. This new node now becomes the parent's child and reduces the number of disambiguations that need to be resolved at that one branching point. For example, the ACE entity type hierarchy has the following children of the implicit "root" node: FACILITY, GPE, (physical) LOCATION, ORGANIZATION, PERSON, VEHICLE, and WEAPON. Since *where* questions could apply to FACILITY, GPE, and LOCATION answer types, these nodes are moved under an abstract LOCATION node and the ACE type is re-named to PHYSICAL-LOCATION for clarity.

Step 3: Separate. When a node could belong in more than one parent nodes, one may create two answer type nodes that point to the same entity type. A obvious concern is whether a UNIVERSITY belongs under ORGANIZATION (*Which university was awarded the grant?*) or LOCATION (*Where did she decide to go to school?*). The solution is to create both a UNIVERSITYORG and UNIVERSITYLOC type. The creators of the ACE hierarchy faced a similar dilemma of where to place COUNTRY, CITY, etc., that had traditionally been considered LOCATIONs but could also be used as an ORGANIZATION. Their solution of adding an extra type as a sibling made classification of extracted entities easier, but it does not work well for a hierarchical disambiguation in a deep hierarchy as the sub-trees of both candidate parents would have to be duplicated.

Step 4: Repeat. Return to both Step 2 and 3 until they no longer apply.

4. Annotating the Question Corpus

In this section, we describe how we used the large ATH introduced in Section 3 in order to annotate a corpus drawn from more than 10,000 questions compiled from (1) existing annotated question corpora (Li and Roth, 2002), (2) collections of questions mined from the web, and (3) questions submitted to LCC's FERRET question-answering system (Hickl et al., 2006a). (A breakdown of the number of questions obtained from each of these three strategies is provided in Table 3.)

Question Set	# Questions
UIUC Train & Test	5,952
Web Crawl	3,485
FERRET log	563

Table 3: Distribution of 10,000 annotated questions by originating data set.

4.1. Methodology

We used a custom graphical user interface (GUI) in order to facilitate annotation of questions with their EATs. Our system presented annotators with individual questions and allowed to select from amongst a set of answer types (provided via a drop-down list).[2] Annotators were tasked with annotating 1000 questions per session and were allowed to work at their own pace (without a time limit) during an annotation session. Each set of 1000 questions were annotated by a team of 2 annotators; the percentage of inter-annotator agreement (as measured by the total number of questions that each pair of annotators associated with the same EAT) was measured after each round of annotation.

[2] A copy of our annotation GUI can be obtained by contacting the authors.

The two major and competing concerns with annotating a large number of answer are the accuracy of the annotations (as measured in inter-annotator agreement) and the time spent annotating. We experimented with three different annotation methodologies as part of this work. Under the first methodology (Methodology 1), annotators were asked to assign the "finest" answer type appropriate for each question. While we anticipated that this process would involve the fewest iterations, it was expected that annotators would have difficulty accurately annotating the answer types due to the size of the hierarchy. In contrast, with Methodology 2, annotators performed annotation using a two-pass approach designed to minimize the total number of disagreements between annotators. Under the first pass, annotators associated each question with one of 11 "coarse" EATs; in a second pass, they then were asked to select which of the "fine" EATs associated with each "coarse" type was most appropriate for the question. With Methodology 3, annotators followed a similar strategy as with Methodology 2: under this approach, however, they were asked to annotate each level of the hierarchy at a time, assigning either a child node of the last assigned answer type or a STOP option if further classification over-specifies the expected answer type. (A summary table of these three methodologies is presented in Table 4.)

Methodology 1. A traditional one-pass annotation where the annotator assigned the final answer type, followed by an inter-annotator agreement session.
Methodology 2. A two-pass annotation where the annotator first assigned a "coarse" answer type (see Table 5), followed by an inter-annotator agreement session. For the second pass, the annotator assigned the final answer type, followed by a final inter-annotator agreement session.
Methodology 3. A multi-pass annotation where a pass was made for each decision point in the hierarchy, with only the immediate children as well as a STOP option available to the annotator. Each session was followed by an inter-annotator agreement session.

Table 4: Methodologies used for annotating answer types.

UIUC Coarse Type	Methodology 2 Coarse Type(s)
ABBREVIATION	ABBREVIATION
ENTITY	OTHER-ENTITY, OTHER-VALUE, WORK
DESCRIPTION	COMPLEX
HUMAN	HUMAN, TITLE
LOCATION	LOCATION, CONTACT-INFO
NUMERIC	NUMERIC, TEMPORAL

Table 5: Coarse types used in Methodology 2.

4.2. Inter-Annotator Agreement

Our initial expectation was that Methodology 1 would be less successful as it would require the annotators to have knowledge of the entire hierarchy, while Methodology 3 would result in the least disagreement, but would take far more time, reducing the amount of data that could be annotated. Methodology 2 was introduced as a compromise between the two – allowing for a single inter-annotator agreement session in the very middle in order to resolve any major differences, to determine if the finer answer types would be easy to annotate once the coarse type is agreed upon.

The actual inter-annotator agreements for all three trials are shown in Table 6. These results fulfill our expectations. Methodology 1 proved very difficult for annotators, as they failed to agree for a majority of questions. Methodology 3 had consistently high agreements, while Methodology 2 agreed with expectation of falling somewhere in between.

Methodology 1	Methodology 2	Methodology 3
47.4%	86.2%	85.3%
		84.7%
	59.4%	91.5%
		97.0%
		99.8%

Table 6: Inter-annotator agreement from all three trials. Each row represents a single pass through the question set.

Comparing inter-annotator agreement between experiments that use a different number of passes and a different number of possible classes is difficult. Therefore, we also calculated the percentage of questions that had disagreement through *any* pass of the data. In other words, if in any pass (up to five for Methodology 3) a particular question had some disagreement between the analysts, then it would count against what we call "question agreement".

Table 7 shows the question agreement for all three passes. Methodology 3 had a much higher overall question agreement than both Methodology 1 and 2. This suggests that Methodology 3 makes a larger percentage of questions easier to annotate. Since the first pass in Methodology 3 has the same candidate answer types as the first pass in Methodology 2, the second pass for Methodology 2 (going from coarse class to final class) must still prove difficult for annotators.

	Percentage
Methodology 1	47.4%
Methodology 2	53.2%
Methodology 3	70.0%

Table 7: Percentage of unique questions without any disagreement through all annotation passes.

5. Answer Type Detection

In this section, we show that current classification-based approaches to ATD can be extended to the task of learning large answer type hierarchies, such as the ones we have introduced in this paper. We compare the performance of two different classification-based approaches to ATD – one based on a relatively "flat" ATH and one based on the more articulated "deeper" ATH we have introduced – on an annotated corpus of 10,000 questions assembled for this work.

5.1. Answer Type Classification

While Section 4 showed that a deep hierarchy can improve inter-annotator agreement, we believe that a well-organized ATH can also improve a system's ability to classify a question's expected answer type. We utilized two methods to experiment with how a hierarchical classifier compares to a baseline flat classifier. Initial results support our hypothesis that a hierarchical approach can improve classification accuracy and that, even with over 200 answer types, such a classifier is comparable to state-of-the-art techniques on smaller hierarchies.

Flat classifiers can provide a useful baseline because of their simplicity and lack of error propagation. However, as the number of outcomes increase, a flat classifier often has difficulty separating among different outcomes, especially when the distribution of types is unevenly distributed as in answer type detection.

A hierarchical classification-based approach to ATD can be thought of as having either a machine or heuristic classifier for every node in the hierarchy that has at least one child. Each classifier may choose from among the child nodes and the parent node (essentially the equivalent of the STOP option from Section 4). The classification process starts from the root node in the hierarchy, gets an initial classification from the root node's children, then proceeds to the corresponding node to re-classify based on its children. This process proceeds until a leaf node is reached or the classifier outcome is the STOP option. A summary of both experiments may be found in Table 8.

Experiment 1: Flat Classification. A completely flat structure (i.e., all known (annotated) outcomes were considered by the classifier) to make a single machine-learning based classification of the final answer type.
Experiment 2: Hierarchical Classification. A hierarchical structure (i.e., only the children of the current type are considered as outcomes) to make a classification at every branching point in the hierarchy. 3 classifiers are machine-learning based while the remainder are heuristic classifiers.

Table 8: Experiments used for classifying answer types.

In a departure from previous machine-learning based approaches (Li and Roth, 2002; Krishnan et al., 2005), we used a maximum entropy classifier to learn our ATH. Our classification process currently uses three machine-learned classifiers. The first resolves all questions into one of 11 "coarse" types that are similar to the UIUC coarse types in Table 1. If the first classifier's outcome is HUMAN, then a machine classifier resolves between INDIVIDUAL, HUMAN-GROUP, ORGANIZATION, and HUMAN (not enough information). If the first classifier's outcome is LOCATION, then a machine classifier is used to resolve between PHYSICAL-LOCATION, GPE, FACILITY, and LOCATION (again, when not enough information is present to make a decision). At every other decision point in the hierarchy, a heuristic classifier uses features extracted from the question to make a decision.

Both experiments were trained on 8,000 and tested on 2,000 of the questions from the annotated set of questions from Section 4. We used the question set annotated with Methodology 3 as we felt the inter-annotator agreement results suggest this question set offers the highest quality data.

Performance on the classification experiments is given in Table 9. Confirming our expectations, the hierarchical approach out-performed the flat classification by almost 7%.

Classification Type	Accuracy
Flat	79.8%
Hierarchical	86.7%

Table 9: Performance of Answer Type Detection Strategies.

We see two main benefits to hierarchical classification that could account for such a score increase.

First, using a multi-pass machine classification approach has the additional benefit of allowing for different feature sets to be used. For example, resources such as WORDNET (Miller, 1995) may be used to aid in the classification of geo-political entities (GPE) (e.g., knowing that a *hamlet* should map to the answer type CITY). Only a simple "is a hypernym of location" feature is necessary in order to decide if a hamlet type from among a coarse set such as {LOCATION, TEMPORAL, PERSON-ORG, NUMERIC}. But a more complicated hypernymy feature is necessary to differentiate a hamlet from other GPEs such as STATE and COUNTRY. By splitting out the classification into multiple layers, the higher-level classification tasks will be benefited through the use of simpler features.

Second, an additional benefit to classification is the hierarchy's property that the final answer type does not need to be a leaf node. For the easier or less important classifications, this allows a simple manual heuristic to make the classification. For example, if a question is currently classified as an ATHLETE question, whose possible children are BASEBALL-PLAYER, FOOTBALL-PLAYER, SOCCER-PLAYER, and SWIMMER, the question's classification may be left at ATHLETE if the heuristic does not fire. This might mean that a question such as *Who is famous for swimming the English Channel?* will not be classified into its finest possible type, but it could very well be determined that writing a machine classification system for the ATHLETE sub-types is not feasible. Furthermore, it has been our experience that for the finest entity types, a heuristic approach is beneficial, as it overcomes the classifier's natural bias due to a sparse amount of training data.

5.2. Impact on Question-Answering

In this section, we show that access to a large ATH can significantly enhance the performance of an end-to-end factoid question-answering system.

Choosing the top answer from a large set of candidate has proven to be a difficult task in question-answering. Several methods exist that could theoretically solve this problem, such as theorem provers and textual entailment systems, but these solutions suffers significantly from both precision and recall issues. A far more viable and less limiting option is to significantly reduce the number of candidate answers. By using a deep answer type hierarchy with an equally detailed set of entity types, the set of candidate answers may be reduced by several factors, which will improve the results of even a basic ranking technique.

We have incorporated the results of our wide-coverage answer type detection system into the factoid question-answering pipeline implemented in LCC's FERRET question-answering system.

First evaluated in the 2005 TREC Question-Answering Evaluations (Harabagiu et al., 2005), FERRET leverages a wide range of lexico-semantic annotations (including output from LCC's (1) syntactic and (2) semantic dependency parsers, (3) LCC's CICEROLITE named entity recognition system and (4) PINPOINT temporal and spatial normalization systems) in order to retrieve and extract exact answers to both factoid and complex questions submitted by users.

Table 10 details the improvement of using the flat set of 11 coarse answer types found in Table 5 against using LCC's full expected answer type hierarchy of more than 200 types.

# of Types	Final Score
11	31.28%
350	38.64%

Table 10: Initial end-to-end Top 1 score of the FERRET Q/A system on 188 TREC 2007 Questions.

Further expansion of the ATH should yield higher end-to-end scores. Question-Answering systems, however, are limited by far more than the number of answer types it is able to classify. Errors in question annotation, document retrieval, named entity recognition, and answer ranking all put an upper bound on the effectiveness of a theoretically infinitely sized ATH. Additionally, there are many different strategies for Q/A, many that require no answer type. The results in Table 10 use a baseline entity-based answer extraction framework. Combining the results of this strategy with others can produce a much more accurate answers.
However, 200 answer types is by no means the limit of useful answer types and it is reasonable that significant performance gains are quite possible. Furthermore, both ATHs discussed in this paper are designed for TREC questions. The set of TREC questions and their corresponding expected answer types by no means represents a true open-domain set. Indeed, the ability to adapt the hierarchy described in Section 3 to include alternate domains such as biological and aeronautical questions means there is no functional limit on the number of answer types.

6. Conclusions

This paper described the creation of a new answer type detection system capable of recognizing more than 200 different expected answer types with greater 85% precision. In a departure from previous work in answer type detection (Krishnan et al., 2005; Li and Roth, 2002), we have demonstrated how a large, multi-tiered answer type hierarchy can be created which incorporates many of the entity types included in LCC's wide coverage named entity recognition system, CICEROLITE; this hierarchy was then used in order to create a new corpus of more than 10,000 questions which could be used to train an ATD system. We showed that the hierarchy we have developed can enhance a state-of-the-art question-answering system (Hickl et al., 2006a) by more than 7% overall.

7. Acknowledgments

The authors would like to thank Bryan Rink, John Williams, Ying Shi, Tobias Jungen, and Jeremy Bensley for their assistance with the annotation experiments described in this paper. This material is based upon work funded in whole or in part by the U.S. Government. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not reflect the views of the U.S. Government.

8. References

Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunsecu, Roxana Girju, Vasile Rus, and Paul Morarescu. 2000. FALCON: Boosting knowledge for answer engines. In *Proceedings of the 9th Text REtrieval Conference*.

S. Harabagiu, D. Moldovan, M. Pasca, M. Surdeanu, R. Mihalcea, R. Girju, V. Rus, F. Lacatusu, P. Morarescu, and R. Bunescu. 2001. Answering Complex, List and Context Questions with LCC's Question-Answering Server. In *Proceedings of the Tenth Text REtrieval Conference*.

S. Harabagiu, D. Moldovan, C. Clark, M. Bowden, A. Hickl, and P. Wang. 2005. Employing Two Question Answering Systems in TREC 2005. In *Proceedings of the Fourteenth Text REtrieval Conference*.

Andrew Hickl, Patrick Wang, John Lehamnn, and Sanda Harabagiu. 2006a. Ferret: Interactive Question-Answering for Real-World Research Environments. In *Proceedings of the 2006 COLING-ACL Interactive Presentations Session*.

Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Ying Shi, and Bryan Rink. 2006b. Question Answering with LCC's Chaucer at TREC 2006. In *Proceedings of the Fifteenth Text REtrieval Conference*.

Andrew Hickl, Kirk Roberts, Bryan Rink, Jeremy Bensely, Tobias Jungen, Ying Shi, and John Williams. 2007. Question Answering with LCC's Chaucer-2 at TREC 2007. In *Proceedings of the Sixteenth Text REtrieval Conference*.

V. Krishnan, S. Das, and S. Chakrabarti. 2005. Enhanced answer type inference from questions using sequential models. In *Proceedings of EMNLP*.

X. Li and D. Roth. 2002. Learning question classifiers. In *Proc. the International Conference on Computational Linguistics (COLING)*.

George A. Miller. 1995. WordNet: a lexical database for English. *Communications of the Association for Computing Machinery*, 38(11):39–41.

Christopher Pinchak and Dekang Lin. 2006. A Probabilistic Answer Type Model. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages pages 393 – 400.