Abstract
Question Classification is an important task in most Question Answering systems. The entire passage retrieval and answer extraction process relies on having the correct answer type for a given question. Much research in this area has focused on employing regular expressions, hand-written grammar rules, and more advanced natural language processing techniques to parse the question and determine what is being asked for. Although this has been successful, the results are not perfect, and furthermore enhancing these systems for new question types is difficult and time consuming. I apply language-modeling techniques to this problem, using entity-tagged language models to classify questions according to answer type. This technique is fully automatic and requires no hand-written rules. Language models are built for each question class, and new questions are classified using these models. I compare my results with those of a Support Vector Machines question classification baseline. I also compare the language modeling technique with others presented in the literature. Throughout, I examine the effects of using very specific answer types.

1 Introduction
In question answering systems, the first step taken is usually to process the question and to determine what it is asking for. In almost all cases, this means choosing the words or features to use in the query and determining the type of the answer. For instance, in the question “Who framed Roger?”, we are looking for an answer of type person name. After a number of documents or passages have been retrieved, the answer is extracted from the retrieved text. This involves finding the entities of the same type as the desired answer, among other things (Nyberg, 2002; Pinto, 2002). Clearly, it is important to have the right answer type. Thus, question classification (QC) is an important task in question answering systems.

Much of the research done on question classification has focused on employing regular expression and hand-written grammar rules to parse the question and determine what is being asked for (Van Durme, 2003). Although this has been successful, there are a number of limitations (Li, 2002; Hacioglu, 2003). First, it is very time consuming. Because these rules are written by hand, we cannot just learn what we need to perform classification from examples. Secondly, these rules are often brittle. That is, when new question types arise, the system is not able to handle them appropriately. New rules must be written by hand to cover these new types. Thirdly, if it is decided that we need to use a new set of answer types, it is probable that many rules will need to be rewritten. Finally, it is even more difficult to extend the answer types to more specific types. As we will see later, most systems with hand-written rules (and even many without) use less than ten answer types. Recently, researchers have become interested in using more specific types, numbering around fifty.

Because of the limitations with hand-written rules, the project described in this paper uses completely automatic techniques to learn how to classify questions. Language modeling techniques are applied to this problem; entity-tagged language models are used to classify questions according to answer type. All that is needed for this method is a set of questions classified according to answer type to be used for training and testing.

In the following sections of this paper, I will detail previous work on question classification and named-entity recognition. I will also
discuss the performance of my technique with respect to a Support Vector Machines baseline, and compare the technique with other existing techniques.

2 Previous Work

This section details previous work on named-entity recognition and question classification.

2.1 Named-Entity Recognition

A named entity is a person, place, etc. that has a specific name. For instance, Microsoft and George Bush are both named entities. The former is a company, or organization, while the latter is a person name. Finding and marking named entities in a piece of text such as a question is an information extraction problem. This problem has been researched a good deal. In this project, we make use of a named-entity recognizer called IdentiFinder, described in (Bikel, 1999).

As you will see, after finding these named entities, we can use them by replacing the entities themselves with the names of the entities. For instance, a phrase such as “I bought the software from Microsoft” would be changed to “I bought the software from <company>.” In our case, questions, we might have questions such as “Who is George Bush?” changed to “Who is <person-name>?”.

2.2 Question Classification

Previous work in question classification pertinent to this study has fallen into two categories. First, there are a number of classification strategies that have been used. Secondly, there has been work on designing appropriate answer types to be used during classification.

2.2.1 Classification Strategies

Besides regular expressions and hand-written rules, there are two main methods of performing question classification. Both are probabilistic methods. The first is using standard machine-learning algorithms. The second is language modeling. I will first introduce the previous work in using machine learning algorithms for question classification and then discuss the language modeling techniques.

The primary machine-learning algorithm used for question classification today is Support Vector Machines, or SVM (Hacioglu, 2003; Zhang, 2003). However, other learning architectures, such as the SNoW learning architecture, are also used (Li, 2002). Question classification precision using just the words of the questions is usually around 50%. However, many features besides words are often used in these classifiers. Named entities are routinely used, as are part of speech tags. Text n-grams, syntax, and semantics are also often used. For instance, one system uses the following as features: the words themselves, POS tags, named-entity tags, head chunks (first noun chunk in a sentence), and semantically related words (words that often occur with a specific question class) (Li, 2002). In this case, these features are only created semi-automatically, because the related words features were constructed partially by hand. However, this is still simpler than hand-writing rules, and most machine-learning question classification systems use only automatically collected data.

Precision is greatly improved by using features beyond words. The system just described achieves a peak performance of 84% correct when training on over 5000 questions (Li, 2002). Other systems have achieved 80% and 82% precision with fully automatically collected data (Zhang, 2003; Hacioglu, 2003). All three of these systems are using the finely detailed answer hierarchy described in the next section. This answer type set has 50 types. Other systems, using the simpler answer set of six or seven answer types have achieved over 90% precision using SVM.

The second probabilistic method of question classification is language modeling. The basic idea here is to discover the probability of the question given a question class (Pinto, 2002). This can be viewed similarly to standard language modeling based information retrieval (Ponte, 1998). A language model is created for each question class, built up from all the questions in that class. This corresponds to a document in the database in standard IR. Given this question class language model, which corresponds to a single answer type, we want to discover the probability that the query, a question here, was generated by this
language model. Both unigram and bigram language models have been used for this.

In addition to just the words of the questions, the question class language models can also take into account named-entity information. Pinto, et al., do this by replacing words in the question with their named entities. For instance, “Who is the President of the U.S.?” becomes “Who is the President of the <location>?” (Pinto, 2002). The authors point out that in these types of questions, location is more meaningful than U.S. That is, we are better able to see the similarities between questions of a given class by having named entities like location instead of U.S. In addition to named entities, part of speech information can also be used to replace certain words. The authors choose to replace nouns with <noun>, in order to get questions like “Who is the <noun> of the <location>?”. They point out that information is both gained and lost by doing this, and so they build models with and without part of speech information.

Performance of unigram models with named-entity tagging was found to be around 74%. Bigram models with named-entity tagging were found to have about 73% precision. Models using part-of-speech tags performed worse, but they were able to join the three best models (unigrams with named-entities and both bigram models) together using a variation of a Borda Count to achieve precision of 75% (Pinto, 2002). In addition, the researchers were able to combine the language models with a regular expression classifier, achieving 81%. The final classifier chose the answer class returned by the regular expression classifier that was ranked highest by the language models (Pinto, 2002). Note that in practice the final classifier has closer to 70% precision.

Also note that these experiments were done using the smaller answer type set, composed of seven types. See the results section for my experiments with language modeling techniques and the finer grained answer taxonomy.

2.2.2 Answer Types

Much of the early work in question answering has used a small number of answer types. Most systems used six or seven types, including person, location, organization, date, quantity, duration, and ordinal. Recently, researchers have become more interested in creating better answer classes, with more fine-grained information. One such set of answer classes was discussed in (Li, 2002). Because of interest from the JAVELIN group in a more detailed answer hierarchy, and because of the acceptance of researchers in question answering, I decided to use this answer set. The hierarchy contains 6 course classes and 50 fine classes. The six course classes, similar to the classes previously used in QC, are the following: abbreviation, entity, description, human, location, and numerical value. The rest of the hierarchy is detailed in Table 1 below. Note that these classes are semantic classes. They are designed to be able to be semantically separable, as opposed to previous conceptual taxonomies, to work better with named-entity recognizers (Li, 2002).

<table>
<thead>
<tr>
<th>Course Type</th>
<th>Fine Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbreviation</td>
<td>Abbreviation, Expansion</td>
</tr>
<tr>
<td>Description</td>
<td>Definition, Description, Manner, Reason</td>
</tr>
<tr>
<td>Human</td>
<td>Description, Group, Individual, Title</td>
</tr>
<tr>
<td>Location</td>
<td>City, Country, Mountain, Other, State</td>
</tr>
<tr>
<td>Numerical Value</td>
<td>Code, Count, Date, Distance, Money, Order, Other, Percent, Period, Speed, Temperature, Size, Weight</td>
</tr>
</tbody>
</table>

Table 1: Answer Types Used

The designers of this answer type hierarchy also point out that the classifications of some questions may be ambiguous using this hierarchy. To avoid this problem, the classifier they use is permitted to give multiple labels
when it is not sure. Thus, multiple answer types can be looked for during the answer extraction step. This helps to make sure answers are not discarded because the classifier only outputted one of the valid classes. Although this is useful in practice, I did not do this when doing my experiments, because we would like to use a strict measure of correctness. In the experiments described here, the precision is the percentage of questions correctly classified using only a single answer label.

3 Experimental Techniques and Baseline

For my experiments, I use an SVM-based classifier as my baseline. I implemented this classifier using LIBSVM, a free library for Support Vector Machines (Chang, 2001). The results of the classifier are discussed in the next section.

For the language modeling classifiers, I make use of the Lemur Toolkit for IR and Language Modeling (Ogilvie, 2002). My techniques for performing retrieval are the same as in (Pinto, 2002). That is, I use simple unigram language models for the question classes, and each class is scored by the probability that it generated the query question. Given question class $C$, the probability of generating question query $Q$ is:

$$P(Q|C) = P(w_1|C) \times P(w_2|C) \times \ldots \times P(w_n|C),$$

where the $w$'s are the words of the query.

I also use Identifinder to tag named entities (Bikel, 1999). The tagged data is used in both the baseline SVM classifier and the language modeling classifiers.

In my preliminary testing, I used a training set of 500 TREC questions and a testing set of another 500. Of these, 500 were already classified, and I classified the other 500 questions by hand. When I began testing, I realized this test set was too small, because performance was poor. I then moved to the larger data set that the results below are based on. This freely available data set had 5500 classified questions for the training set, and 500 more for testing. The 500 for testing are the set from the Question Answering Track of TREC 10. The other 5500 are built from previous TREC questions as well as from users of online question answering systems (Li, 2002).

4 Results

In order to test the effects of using entity-tagged language models, I built three language modeling classifiers and another three SVM classifiers. The first classifier of each type used only the words of the questions. The second classifier used the words of the questions plus the named entities. The third classifier replaced words with their named entities, so it used a percentage of the original words and all of the named entities. I decided to try the second classifier because, although this was not tested by the previous language modeling classifiers, it is done in machine learning techniques. That is, in machine learning classifiers, words are not replaced with their named entities. The named entities are simply added in as additional features. It is of course possible that the words with named entities could be filtered out during feature extraction if they do not occur often, but this is not explicitly done.

The first SVM classifier, using words only, performed worse than the first language modeling classifier. The SVM classifier had 44% precision, using the fine-grained answer set of 50 types. The equivalent language modeling classifier had 48% precision. Adding named entities brought the SVM classifier up to 51% precision, and replacing the words with named entities, the third classifier, also had 51% precision. The SVM classifier correctly classified the same number of questions in both cases. It clearly improved by making use of the named entities, but having the original words for those entities neither helped nor hurt performance. The language modeling classifier, on the other hand, had a small difference between the second and third methods. The addition of named entities resulted in 53% precision, and the replacement of words with named entities resulted in 54% precision. See Table 2 for all of these results.

The language modeling classifier seems to perform slightly better than the SVM classifier. However, the performance of the language modeling classifiers are lower than the performance of the language modeling classifiers seen earlier (Pinto, 2002). The earlier classifiers were using a courser version of the answer set, with only seven types. Thus,
it would be beneficial to compare my language modeling classifiers with the earlier ones using the courser version of this answer set. When allowing a correct answer to be any subtype of the correct course type, the precision jumps to 62% and 65%, for the addition of named entities and the replacements of words with named entities, respectively. The performance of the SVM classifier is also improved in this case. See Table 3 for all of these results.

<table>
<thead>
<tr>
<th></th>
<th>SVM Classifier</th>
<th>Language Modeling Classifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Words Only</td>
<td>44%</td>
<td>48%</td>
</tr>
<tr>
<td>Plus Named Entities</td>
<td>51%</td>
<td>53%</td>
</tr>
<tr>
<td>Replace Words with Named Entities</td>
<td>51%</td>
<td>54%</td>
</tr>
</tbody>
</table>

Table 2: Efficient for fine-grained answer set

<table>
<thead>
<tr>
<th></th>
<th>SVM Classifier</th>
<th>Language Modeling Classifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Words Only</td>
<td>50%</td>
<td>51%</td>
</tr>
<tr>
<td>Plus Named Entities</td>
<td>60%</td>
<td>62%</td>
</tr>
<tr>
<td>Replace Words with Named Entities</td>
<td>62%</td>
<td>65%</td>
</tr>
</tbody>
</table>

Table 3: Efficiency for course-grained answer set

Although the precision of my language modeling classifiers did not quite reach the precision of the previously discussed language modeling classifiers, which were around 74%, this is still reasonable. Because I am using a different data set and a different set of answer types, the performance is not expected to be identical. The next section does describe possibilities for increasing performance, however.

The SVM classifiers are also not doing as well as those described earlier, which were around 80% precision. The difference between the 60% precision here and the 80% precision there is significant. However, those reaching 80% precision usually use extra features like part of speech and chunking information. These features can be added straightforwardly to these classifiers to improve performance.

5 Future Work

Performance is still an issue. The SVM classifier can clearly be improved in the same ways as that of the previously described machine learning techniques. However, improving the language modeling classifiers is not as straightforward. It is not immediately clear how some of the features applicable to improving the SVM classifier, such as chunking, could be used to improve the language modeling classifiers. Some features like part of speech and semantically related words fit well though.

Another aspect of future work has to do with the named-entity tagging. Certain named entities, such as GPE (geographic and political entity), replace words like city and country, which should not be replaced with the same word. Questions that are asking for a city name fall into a different category than those that ask for a country name, but if both are replaced with GPE, we lose this information. Thus, it would be worthwhile to determine exactly which named entities are useful in these conditions and which should not be used.

It would also be useful to study different methods of smoothing to determine which method is better for these question classes. The frequency of terms in these classes is significantly different than that of normal documents, so tests would need to be performed on more similar data sets.

One final item that should probably be looked at more in the future is combining language modeling with other classifiers. We have seen in (Pinto, 2002) that performance can be improved by combining the language modeling classifiers with a regular expression model, because each method classifies different kinds of questions correctly. It would be worthwhile to perform a detailed analysis of the types of questions various classifiers are good at classifying correctly, and attempt to leverage this information to improve performance.
6 Conclusion

Overall, the language modeling classifiers performed well with respect to the SVM baseline. Precision was close to that which was previously achieved on a different dataset and answer type set.

The language modeling question classification scheme also extended well to the finer answer taxonomy with 50 types. Performance degraded, but it is evident that the language-modelling scheme has the potential to use more detailed answer taxonomies.

I have also mentioned a number of items in Future Work that should lead to additional improvements.

References


