Question Type Classification Using a Part-of-Speech Hierarchy

Richard Khoury

Department of Software Engineering, Lakehead University,
955 Oliver Road, Thunder Bay, Ontario, Canada, P7A 5E1
Richard.Khoury@lakeheadu.ca

Abstract. Question type (or answer type) classification is the task of determining the correct type of the answer expected to a given query. This is often done by defining or discovering syntactic patterns that represent the structure of typical queries of each type, and classify a given query according to which pattern they satisfy. In this paper, we combine the idea of using informer spans as patterns with our own part-of-speech hierarchy in order to propose both a new approach to pattern-based question type classification and a new way of discovering the informers to be used as patterns. We show experimentally that using our part-of-speech hierarchy greatly improves type classification results, and allows our system to learn valid new informers.

Keywords: Natural Language Processing, Question Type Classification, Answer Type Classification, Part-of-Speech, Syntactic Pattern, Informer Span.

1 Introduction

Question type (or answer type) classification is the Natural Language Processing (NLP) task of determining the correct type of the answer expected to a given query, such as whether the answer should be a person, a place, a date, and so on. This classification task is of crucial importance in question-answering (QA) systems. Indeed, correctly pinpointing the expected answer type of a question allows a QA system to use type-specific answer retrieval algorithms and to reject possible answers of the wrong type [1]. In fact, it has been shown that questions that have been classified into the correct type are answered correctly twice as often as misclassified questions [1]. Over the years, many varied approaches to question type classification have been proposed. A large proportion of systems, including many of those entered in the TREC QA competition, simply try to accomplish this task in one of two ways, either by detecting keywords in the query [1] or by relying on the wh-term (who, what, where, when, why, which, whom, whose, how) at the beginning of the query for disambiguation. However, both approaches are too simplistic and can be easily misled. Keywords alone are not enough to differentiate between question types: for example, the query "who was the French emperor defeated at Waterloo" and "when was a French emperor defeated at Waterloo" will have the same keywords after stopword removal, but belong to different types. And wh-terms can take different meanings based on the context. For example, "who was Napoleon" and "who

M. Kamel and F. Karray (Eds.): AIS 2011, LNAI 6752, pp. 212–221, 2011.

defeated Napoleon" both begin with *who* but only the second is asking for a person while the first is asking for a historical definition, and "what French emperor was defeated at Waterloo" is a *who* query phrased as a *what* query. In fact, research has shown that rephrasing a question to use a different wh-term is the single most common way that humans use to paraphrase queries [2]. In light of this, syntactic-pattern-based methods have become popular in question type classification systems [3]. These methods use or discover syntactic patterns that represent the structure of typical queries of each type, and classify a given query according to which pattern they satisfy. For example, a system could discover, by comparing example queries, that those featuring the pattern "...in which year..." should be classified in the *time* type [3].

In this paper, we combine an idea from Krishnan *et al.* of using informer spans as question type classification patterns [4] with our own part-of-speech hierarchy [5] in order to propose both a new approach to pattern-based question type classification and a new way of discovering the informers to be used as patterns. We show experimentally that using our part-of-speech hierarchy together with a few simple informer spans greatly improves type classification results, and moreover that the hierarchy can be used to learn new valid informers from example queries.

The rest of this paper is organized as follows. In Section 2, we review a representative sample of work done on the task of question type classification in order to clearly illustrate the nature of our contribution. The theoretical frameworks of our classification system, of our learning algorithm and of our part-of-speech hierarchy are all presented in Section 3. Our ideas have all been implemented and tested, and experimental results are presented and discussed in Section 4. Finally, we offer some concluding remarks in Section 5.

2 Background

Question type classification is an integral task in most QA systems developed today. Consequently, there are considerable variations in the nature of the classification systems, in the set of question types recognized, and in the nature and size of the knowledge base underlying the classification systems.

A purely Bayesian solution to the type classification problem was proposed in [6]. The authors used a straightforward Naïve Bayes classifier, and computed the probability of a query belonging to a question type given the prior probability of that question type multiplied by the conditional probability of the query's features (i.e. the words left after stemming and stopword removal) given that type. They used six question types for their classification, namely time, location, human, number, object, and description. Their system achieves an average precision of 59% over these six types; however the authors do not discuss how large the table of conditional probabilities has to be to account for a reasonable number of features in six categories.

A team from the University of Concordia developed a simple keyword-based question type classification system as part of their QA system for the TREC-2007 competition [1]. Their system classifies queries into seven types, namely date, location, person, organization, number, website, and other, by recognizing keywords

in the query. The authors do not give the size of the lexicon they manually built over the years for that purpose, but in [1] they report adding 50 new words into it. Their research does show that a keyword-matching approach can work quite well given a large and detailed enough lexicon: they report that their system achieves 93% accuracy. However, a smaller and coarser lexicon is not necessarily useful, and can even confuse a system, as indicated by Tomuro in his study on the impact of semantic information in question type classification [7]. Tomuro built two question type classifiers, one using a decision tree and one using a k-nearest-neighbour algorithm, and trained two instances of each classifier. The first instance used only a closed lexicon of about 100 words frequently found in queries. This lexicon is thus composed mostly of domain-independent, non-content, closed-class words and includes wh-terms. The second instance of both classifiers uses the same lexicon and also adds in the categorization (i.e. WordNet hypernym) of query words that are not part of the lexicon. The classifiers were trained to recognize 12 question types, namely time, location, entity, definition, reference, reason, procedure, manner, degree, atrans, interval, and yes-no questions. His results show that taking the extra keywords into account and adding the categorization information does not affect the results in a statistically significant way. Moreover, he found that the extra knowledge added can mislead the classifier when a query's syntax affects its semantic meaning. For example, the query "what does Hanukah mean?" is clearly a definition-type question, but the keyword "Hanukah" is a hyponym of "time period" in WordNet, and thus misleads his system into classifying it as a time-type question [7]. Another WordNet-based question type classifier proposed in [8] faced the same difficulties, and overcame them by creating a two-step classification scheme. It begins by defining a set of simple question type syntactic patterns, such as "what {is|are} <phrase>?" for definition-type queries. Their system attempts to match queries to these patterns, and then goes to WordNet hypernym searches for queries that are not recognized by them. The results in [7] and [8] illustrate how syntactic patterns can be more powerful than simple keyword recognition for question type classification, as they offer a more complete picture of the query.

Zhang and Nunamaker [9] built a pattern-based question type classifier for their video indexing and retrieval system. They defined nine question types in their system, namely time, location, person, organization, number, object, reason, definition, and undefined. Their system classifies each user's query according to a set of simple patterns that combine both wh-term and the categories of keywords found in the query. To illustrate, a sample pattern given in [9] is "if (question starts with 'what' + person) then (answer type is person)". Unfortunately, the authors do not give a complete list of patterns nor the total number of patterns in their system. The authors of [4] take the idea of type classification patterns one step in a different direction and propose that the patterns could consist simply of a short string of contiguous words found in the query, which may or may not include wh-terms, and which they called the *informer span* (or simply *informer*) of the query. Their work shows clearly that a support vector machine classifier using hand-made informers yields better results than question bigrams, and that informers discovered automatically work almost as well as hand-made ones.

We can point out a common thread in these methods, namely their reliance on large knowledge bases. The keyword-based system of [1] requires a massive lexicon

of keywords likely to be observed in queries of each type, and the Bayesian system of [6] further needs to define the conditional probability of each type given each keyword, while the categorization systems of [7] and [8] use the WordNet lexicon to recognize classes of keywords. However, keyword-based question type classification is inherently limited and misleading, since the syntactic structure of the query can change the semantic importance and meaning of its keywords [2], [7]. The systems proposed in [9] and [4] compensate for this problem by developing syntactic patterns instead of keyword lists, but they still suffer from the need to develop massive lists of patterns from which to find the one that exactly matches the query. In fact, Krishman *et al.* point out in their review that such systems are built on hundreds of unpublished patterns [4].

3 Methodology

In this paper, we develop a new system for question type classification based on a part-of-speech hierarchy that we present below. As will become evident, using the part-of-speech hierarchy makes it possible to get good classification results using only a handful of simple informers. This stands in stark contrast to the other systems reviewed in Section 2, which need large sets of patterns, lexicons, or probability tables to work well.

The question types we use are "person" (who), "date" (when), "physical object" (what), "location" (where), "numeric value" (how many/how much), and "description" (how/why). We manually define a set of 14 simple informers to represent these question types; a simple enough task given that they each have clear wh-terms. We design these informers to be two words long each so that none would have a length advantage over the others. There are two informers per question type, except for the description type which has two different wh-terms and therefore four informers. The informers are listed in Figure 1.

Who is	What was	How did
Who was	Where is	How does
When is	Where was	Why did
When was	How many	Why does
What is	How much	

Fig. 1. 14 basic informer spans

3.1 Part-of-Speech Hierarchy

A part-of-speech (POS) is commonly defined as a linguistic category of lexical items that share some common syntactic or morphological characteristics. However, it is telling that, despite the concept of a POS being thousands of years old, grammarians and linguists still cannot agree on what exactly the shared characteristics are. This question has very real practical consequences: depending on where the line is drawn between common characteristics and distinguishing differences, one can end up with anywhere from the eight parts-of-speech defined in English textbooks to the 198 parts-of-speech of the London-Lund Corpus of Spoken English [10].

Our solution to this problem is to organize lexical items not into a single set of parts-of-speech but into a part-of-speech hierarchy. The lower levels of this hierarchy feature a greater number of finely-differentiated parts-of-speech, starting with the Penn Treebank POS tags at the lowest level, while the higher levels contain fewer and more general parts-of-speech, and the topmost level is a single all-language-encompassing "universe" part-of-speech. Another innovation has been the inclusion in our hierarchy of several "blank" parts-of-speech, to represent and distinguish between the absences of different types of words. In total, our hierarchy contains 165 parts of speech organized in six levels. We originally developed it in the context of a keyword-extraction project; a detailed description of the hierarchy can be found in the paper describing that project [5].

We can define the semantic importance of different lexical items by assigning weights on the connections between POS in the hierarchy. This allows us to specialize the hierarchy for use in different NLP tasks. In our earlier work on keyword extraction [5], weight was given to verbs and nouns – the typical parts-of-speech of the keywords we were looking for. For question type classification, however, verbs and nouns are not the most semantically important words to take into account. Indeed, Tomuro [7] has shown that question type classification relies mostly on closed-class non-content words. The semantic weight in our hierarchy was thus shifted to the subtrees corresponding to popular query adverbs and pronouns, including wh-terms, and calibrated using queries from the 2007 TREC QA track [11] as examples. The value of each POS in our hierarchy is then computed on the basis of its semantic weight and of the number of descendents it has. The resulting hierarchy, with the value of each POS, is presented in Figure 2.

We showed in [5] how using a part-of-speech hierarchy makes it possible to mathematically define several linguistic comparison operations. It is possible to compute the similarity between two words or POS simply as a function of their distance in the hierarchy. This computation takes several factors into account, including the number of levels in the hierarchy that need to be traversed on the path from one POS to the other and the value of each intermediate POS visited. Likewise, we can find a general place-holder POS to represent two words simply by finding the lowest common ancestor of both words in the hierarchy, and the similarity of the place-holder compared to the original two words it represents is again a function of the distance in the hierarchy between each word and the place-holder POS. By extension, we can measure the similarity between two sentences by pairing their words together by similarity; the fact that our hierarchy includes "blank" parts-of-speech means that the two sentences do not need to be of the same length to be compared. And finally, we can merge two sentences by pairing their words together by similarity and replacing each pair by its lowest common ancestor in the hierarchy.

It is now straightforward to see how our part-of-speech hierarchy can be used for the task of question type classification. Given a list of informers representing different question types, we can use the hierarchy to compute the similarity between a user-specified query and each informer. The query is then classified to the same type as the informer it is most similar to.

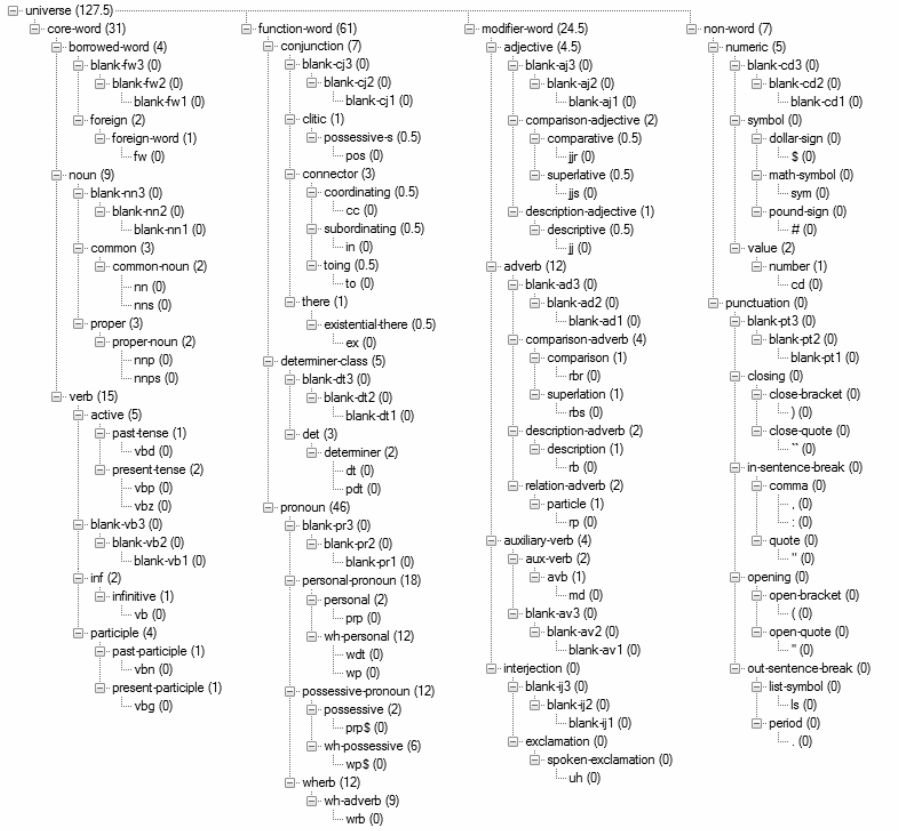

Fig. 2. The POS hierarchy (POS values are in brackets)

3.2 Informer-Learning Algorithm

In addition to being used to compare queries and informers together, our part-of-speech hierarchy can be used as the core of an informer-learning algorithm. The intuition behind the learning algorithm is that, when merging two queries together, irrelevant words will be replaced by high-level POS while informer words common to both queries (if there are any) will remain words or low-level POS in the merged query. We can then extract the informer from the merged query by deleting POS of a level higher than a set threshold and keeping only contiguous words and POS below that threshold.

The informer-learning algorithm we use is summarized in Figure 3. It begins with a set of training queries classified in their correct types and a list of informers representing each type such as the list already proposed in Figure 1. It then divides the set of training queries into two subsets, one containing queries that can be correctly classified by the current informers and the other containing queries that cannot. In Figure 3, we call these sets C-queries and I-queries, for Correctly-classified queries and Incorrectly-classified queries respectively. The learning algorithm then merges

together pairs of incorrectly-classified queries of the same type to generate new informers. Good informers, that can be used to correctly classify some of the incorrectly-classified queries without leading to misclassification of the already correctly-classified queries, are added to the list of informers, and the queries they correctly classify are moved to C-queries, the set of correctly-classified queries. The enriched list of informers is the final result of the learning algorithm.

```
1. Input: list of informers, training queries
2. For each training query
   3. Classify using informers
   4. If classified correctly, add to C-queries
   5. Else, add to I-queries
6. For each pair of misclassified queries of same
type
   7. Merge and generate informer
   8. If informer correctly classifies some of I-
   queries and does not misclassify any of C-queries,
   add to the list of informers
   9. Move newly-correctly-classified I-queries to C-
   queries
10. Return the list of informers
```

Fig. 3. Structure of the learning algorithm

4 Experimental Results

For our experiments, we built a test corpus of queries using the 459 queries from the 2006 TREC QA track [12]. We tagged the words of the queries with their parts-of-speech using the standard Brill tagger, and we manually classified the queries into their correct question types.

Our first experiment is meant to study the classification results obtained by using our 14 basic informers with and without our part-of-speech hierarchy, to show the impact of the hierarchy. Classification using the hierarchy and the basic informers is done as described in Section 3.1. The results without using the hierarchy are meant to be a benchmark. They will show how well a system can perform the classification task by only recognizing the informers in the queries. An initial check shows that the 14 informers are a resource of limited usefulness: they only appear in 43% of our test queries, and in about 12% of these cases they are used in the wrong question type because of the paraphrasing phenomenon studied in [2]. The informers give no useful information to help classify the remaining 57% of queries they do not appear in. This gives insight into the reason why other systems must rely on hundreds of patterns [4].

Next, we used the learning algorithm to expand the list of informers. The set of training queries we used is the query list from the 2007 TREC QA track [11], which we tagged and classified into correct types in the same way as we did for the 2006 TREC queries. The algorithm learned four new informers, which we present in Table 1. In the informers in that table, actual words are written plainly while POS from our

hierarchy are written in square brackets. The words corresponding to the informers in the sample queries are marked in bold.

In each case, we computed the precision and recall of the classification of queries into each of our six question types, using the standard equations given in (1) and (2). We then computed the average precision and recall over all six types, and the average F-measure using equation (3). The results of these three experiments are given in Table 2.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{1}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{2}$$

$$\text{F} - \text{Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

Table 1. New informers learned

Informer	Question type	Sample query
[nn] [common-noun]	person (who)	**Name members** of the group.
[wh-personal] [present-tense]	location (where)	**Which** college did she go to?
what does	description (how/why)	**What does** LPGA stand for?
[nn] is [dt] [nn] ?	description (how/why)	What kind of **animal is an agouti?**

Table 2. Question Type Classification Results

#	Experiment	Precision	Recall	F-Measure
1	Without hierarchy	43%	41%	42%
2	With hierarchy	85%	56%	68%
3	With learning	74%	65%	69%

4.1 Discussion

From the results presented in Table 2, it can clearly be seen that our system (line #2) does much better than the benchmark (line #1). It achieves nearly twice the precision and yields a 26% increase in F-measure. The reason for this improvement, and the only difference between our system and the benchmark, is the use of the part-of-speech hierarchy in addition to the 14 basic informers. The benchmark system can only exactly match queries to the informers; if there is no such match or if several informers appear in a query, the system is clueless. On the other hand, our system compares queries and informers together using the POS hierarchy as described in Section 3.1, and computes the similarity of each pair. In other words, it associates a query to its most similar informer, rather than look for an exact match. This ability to handle similar but inexact matches is clearly an important advantage.

In our third experiment, the learning algorithm discovers four new informers. They represent different syntaxes of queries that were not accounted for in our initial 14

informers, as illustrated in Table 1. In particular, the first informer is learned to handle an error in the tagging: the Brill tagger mistakenly identified the word "name" in these queries as a noun instead of a verb. The next two informers in Table 1 are learned to handle the wh-term parts of varying styles of queries; the second one in particular is a general inexact match in queries made possible by our hierarchy. The fourth informer represents a longer sentence structure often found in description-type queries.

The classification results using our hierarchy and the informer list including the four learned informers (line #3) show an important improvement compared to the benchmark (line #1). However, the advantage is less clear when compared to the 14 basic informers alone (line #2). The new classification shows a worse precision but a better recall, leaving the F-measure nearly unchanged. It is worth noting that using the basic informers alone leads to an important 30% difference between precision and recall, while that gap is reduced to 10% when the extra four informers are added in, giving our system a more balanced classification performance.

Although both experiments using the POS hierarchy outperform the benchmark, there is still clearly room for improvement. Errors in our system are misclassifications caused by an informer from a wrong question type being more similar to a query than any of the informers from its own type. The main bad informer is "what was", one of our 14 basic informers, which is alone responsible for 75% of misclassifications in each of the two experiments with our system. However, that informer cannot be simply eliminated, as it is also responsible for a lot of correct classifications; indeed, more than a third of the queries it classifies are done so correctly. Rather, the solution is for the system to learn new informers that will be more similar to the misclassified queries than "what was" and will classify them correctly. The fact that our current learning algorithm does not discover these informers might be due to our strict learning criterion of only saving informers that do not misclassify any queries from the C-queries subset. A more permissive criterion, for example of accepting informers that correctly classify more of the I-queries subset than they misclassify the C-queries subset, could lead to learning a better informer list. Changes to the learning algorithm in order to discover better informers will be studied in future research.

5 Conclusion

In this paper, we present a new method to learn and apply informers for the task of syntactic-pattern-based question type classification. What sets our method apart is the use of our part-of-speech hierarchy, which makes it possible to compute mathematically the distance between the informers and the queries, and to associate queries with their most similar informers rather than look for exact keyword matches. In fact, experimental results show that using the hierarchy for this task yielded an average 26% improvement in the F-measure of type classification. Another advantage of our method is that it obtains good results using only 14 simple informers, while traditional methods use hundreds of keywords, patterns or probabilities. Future work will focus on further improving the classification by refining the informer-learning algorithm to discover new and better informers, which will account for and rectify some of the mistakes the informers in the current system cause.

References

1. Razmara, M., Fee, A., Kosseim, L.: Concordia University at the TREC 2007 QA track. In: Proceedings of the Sixteenth Text REtrieval Conference (TREC 2007), Gaithersburg, USA (2007)
2. Tomuro, N.: Interrogative reformulation patterns and acquisition of question paraphrases. In: Proceedings of the Second International Workshop on Paraphrasing, Sapporo, Japan, vol. 16, pp. 33–40 (2003)
3. Sung, C.-L., Day, M.-Y., Yen, H.-C., Hsu, W.-L.: A template alignment algorithm for question classification. In: IEEE International Conference on Intelligence and Security Informatics (ISI 2008), pp. 197–199 (2008)
4. Krishnan, V., Das, S., Chakrabarti, S.: Enchanced Answer Type Inference from Questions using Sequential Models. In: Proceedings of Human Language Technology Conference / Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005), Vancouver, Canada, pp. 315–322 (2005)
5. Khoury, R., Karray, F., Kamel, M.: Keyword extraction rules based on a part-of-speech hierarchy. International Journal of Advanced Media and Communication 2(2), 138–153 (2008)
6. Liang, Z., Lang, Z., Jia-Jun, C.: Structure analysis and computation-based Chinese question classification. In: Sixth International Conference on Advanced Language Processing and Web Information Technology (ALPIT 2007), Luoyang, China, pp. 39–44 (2007)
7. Tomuro, N.: Question terminology and representation of question type classification. In: Second International Workshop on Computational Terminology, vol. 14 (2002)
8. Harabagiu, S., Moldovan, D., Pasca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Girju, R., Rus, V., Morarescu, P.: Falcon: Boosting knowledge for answer engines. In: Proceedings of the 9th Text REtrieval Conference (TREC-9), Gaithersburg, USA, pp. 479–488 (2000)
9. Zhang, D., Nunamaker, J.F.: A Natural language approach to content-based video indexing and retrieval for interactive e-learning. IEEE Transactions on Multimedia 6(3), 450–458 (2004)
10. Marcus, M., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of English: the Penn Treebank. Computational Linguistics 19(2), 313–330 (1993)
11. Dang, H.T., Kelly, D., Lin, J.: Overview of the TREC 2007 Question Answering Track. In: Proceedings of the Sixteenth Text REtrieval Conference (TREC 2007), Gaithersburg, USA (2007)
12. Dang, H.T., Lin, J., Kelly, D.: Overview of the TREC 2006 Question Answering Track. In: Proceedings of the Fifteenth Text REtrieval Conference (TREC 2006), Gaithersburg, USA (2006)