Question Classification by Structure Induction*

Menno van Zaanen and **Luiz Augusto Pizzato** and **Diego Mollá**
Division of Information and Communication Sciences (ICS)
Department of Computing
Macquarie University
2109 Sydney, NSW
Australia
{menno, pizzato, diego}@ics.mq.edu.au

Abstract

In this article we introduce a new approach (and several implementations) to the task of question classification. The approach extracts structural information using machine learning techniques and the patterns found are used to classify the questions. The approach fits in between the machine learning and handcrafting of regular expressions (as it was done in the past) and combines the best of both: classifiers can be generated automatically and the output can be investigated and manually optimised if needed.

1 Introduction

Before a question answering (QA) system can answer a question, it needs to have an idea what the question is about. One of the principal tasks of the question analysis stage of a QA system is the determination of the expected answer type (EAT). Finding the EAT of a question is called question classification (or EAT classification) [Hermjakob, 2001].

In this work we introduce a new approach to the problem of EAT classification, based on structural information that can be extracted from the questions. Re-occurring structures found in questions, such as "How far ..." may help finding the correct EAT (in this case "distance") for a particular question. The approach described here automatically finds these structures during training and uses this information when classifying new questions.

Our approach combines the two main methods to question classification: machine learning and pattern matching. Using machine learning, patterns are extracted from the training data. These patterns serve as regular expressions during the classification task.

In the next two sections, we will describe two systems that fit into this approach. The first one uses a grammatical inference system and the second one is based on tries.

2 Alignment-Based Learning Classifier

The structure extraction phase of the first system is done by Alignment-Based Learning (ABL) [van Zaanen, 2002]. ABL is a generic unsupervised grammatical inference framework that learns structure from plain text sentences. The underlying idea of ABL is that constituents can be interchanged. To give an example, if we exchange the noun phrase *the man* in the sentence *He sees the man* with another noun phrase *a woman*, we get another valid sentence: *He sees a woman*. This process can be reversed (by aligning sentences) and possible constituents, called hypotheses can be found. This is called the alignment learning phase, one of the three phases of ABL, and the one that we use in this article. Table 1 shows an example of the structure learned from a toy corpus of 4 sentences.

"DESC"	*(What)(is (caffeine))*
"DESC"	*(What)(is (Teflon))*
"LOC"	*(Where) is (Milan)*
"LOC"	*What (are the twin cities)*

Table 1: Example sentences with ABL structure

In the training phase, regular expressions associated with the structures found are stored together with the EAT of the corresponding questions. Several EATs with their frequencies may be stored if a regular expression matches several questions.

We have experimented with two implementations. The first implementation, which we call *hypo*, uses the words in the hypotheses (i.e. the words between the brackets) to form regular expressions which are stored together with the EATs of the associated questions. The second implementation, called *unhypo*, uses the words that are left after removing each hypothesis. For example, the *hypo* version uses the first question of Table 1 to produce the patterns /What/, /is caffeine/, and /caffeine/, whereas *unhypo* produces the patterns /is caffeine/, /What/, and /What is/.

During the classification phase we have experimented with two further approaches that differ on the use of the frequency counts of the matched regular expressions. The first method (called *default*) increments the counts of the EATs of the question by the frequency that is stored with the EATs of the regular expression. The second method (called *prior*) will simply increment the counts by 1. When all regular expressions are tried, both methods select the EAT with the highest count. If there several expressions with the same count, the

*This work is supported by the Australian Research Council, ARC Discovery Grant no. DP0450750.

default method makes a random choice, whereas the *prior* method selects the EAT with the highest overall frequency.

Finally, we have experimented with the impact of the parts of speech (POS) as a simple approach to disambiguate words. We used Brill's tagger [Brill, 1992] and the resulting POS information is simply combined with the plain words. Adjacent words that have the same POS are combined into one token. For example the question *Who is Federico Fellini?* is, after POS tagging, divided into three tokens: (*Who*, WP), (*is*, VBZ) and (*Federico Fellini*, NNP).

3 Trie Classifier

The second system uses a trie structure. A trie $T(S)$ is a data structure defined by a recursive rule $T(S) = \{T(S/a_1), T(S/a_2), \ldots, T(S/a_r)\}$. S is a set of sequences whose elements are taken from the alphabet A. S/a_n is the set of strings that contains all strings of S that start with a_n but stripped of that initial element [Clément *et al.*, 1998].

During the learning phase, all questions are inserted into a trie structure that contains, in addition to the token, the EAT and frequency information (the number of questions that use that path in the trie).

During the classification phase, the trie is traversed in the usual way. If the new question is a prefix of a training question the traversal is trivial, and the node at the end of the traversal path indicates the EAT of the question.

To find a path for unseen questions, non-matching nodes are skipped in a methodical way in what we call the look-ahead process. Let us say that question tokens of question $Q = q_1 q_2 \ldots q_n$ match up to q_i: $T(S/q_1/q_2/\ldots/q_i)$, and $T(S/q_1/q_2/\ldots/q_i/q_{i+1})$ does not exist. The look-ahead process then builds a set of sub-tries of the form $\{T(S/q_1/q_2/\ldots/q_i/\beta/q_{i+2}) | \beta \in A\}$. The sub-trie with the highest frequency associated is selected, and the process continues with q_{i+2} until all tokens are consumed.

There are two variations of the above process. In the *strict* method, β and q_{i+1} must have the same POS tag. If the resulting set of sub-tries is empty, the search process stops and the EAT is retrieved from the node q_i in the trie. In the *flex* method, if an empty set of sub-tries is retrieved, we consider β as a sequence of question words until we find a question word equal to q_{i+2}.

We also experimented with a set of variations that have no POS information. In this case, β and q_{i+i} must be exactly the same token. Again we allow for a *strict* and a *flex* version.

4 Results

To compare our systems with current machine-learning methods, we have used the same data as [Zhang and Sun Lee, 2003]. This is a collection of 5,452 training questions and 500 test questions. The data have 6 coarse-grained classes and 50 fine-grained classes.

Table 2 indicates the precision of the questions classified with the coarse-grained classes for all our systems and for a baseline that always selects the most frequent class according to the training data. This baseline is, of course, the same for the plain words and POS tagged data. All our implementations performed well above the baseline.

			words	POS
Baseline			0.188	0.188
ABL	hypo	default	0.516	0.682
		prior	0.554	0.624
	unhypo	default	0.652	0.638
		prior	0.580	0.594
Trie	strict		0.844	0.812
	flex		0.850	0.794

Table 2: Results on the coarse-grained data

The results show that the POS information helps the performance of the ABL method but not of the trie-based method. Overall, the trie-based approach outperforms the ABL implementations. We obtained similar results with the fine-grained data (not shown for reasons of space).

Our best results fall close to the best-performing system in the bag-of-words and bag-of-n-grams versions of [Zhang and Sun Lee, 2003]. Their results ranged from 75.6% (Nearest Neighbours on words) to 87.4% (SVM on n-grams). Given the simplicity of our methods and their potential for further improvement, this is encouraging.

5 Conclusion

The results on the annotated questions of the TREC10 data show that the approach is feasible and both systems generate acceptable results. We expect that future systems that fall in the structure induced question classification approach (for example, based on other grammatical inference systems) will result in even better performances.

The automatically learned regular expressions can be inspected and extended by humans. The systems therefore combine the advantages of machine-learning and pattern-matching methods.

Additionally, we think that the structure found by the systems can be used to find the focus of the question.

References

[Brill, 1992] Eric Brill. A simple rule-based part-of-speech tagger. In *Proc. ANLP*, pages 152–155, Trento, Italy, 1992.

[Clément *et al.*, 1998] J. Clément, P. Flajolet, and B. Vallée. The analysis of hybrid trie structures. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 531–539, Philadelphia:PA, USA, 1998. SIAM Press.

[Hermjakob, 2001] Ulf Hermjakob. Parsing and question classification for question answering. In *Proc. ACL/EACL Workshop on Open-Domain Question Answering*, 2001.

[van Zaanen, 2002] Menno van Zaanen. *Bootstrapping Structure into Language: Alignment-Based Learning*. PhD thesis, University of Leeds, Leeds, UK, January 2002.

[Zhang and Sun Lee, 2003] Dell Zhang and Wee Sun Lee. Question classification using support vector machines. In Charles Clarke, Gordon Cormack, Jamie Callan, D avid Hawking, and Alan Smeaton, editors, *Proc. SIGIR*, pages 26–32, New York:NY, US, 2003. ACM Press.