Flexible Answer Typing with Discriminative Preference Ranking

Christopher Pinchak[†] **Dekang Lin**[‡] **Davood Rafiei**[†]

[†]**Department of Computing Science**
University of Alberta
Edmonton, Alberta, Canada
{pinchak,drafiei}@cs.ualberta.ca

[‡]**Google Inc.**
1600 Amphitheatre Parkway
Mountain View, CA, USA
lindek@google.com

Abstract

An important part of question answering is ensuring a candidate answer is plausible as a response. We present a flexible approach based on discriminative preference ranking to determine which of a set of candidate answers are appropriate. Discriminative methods provide superior performance while at the same time allow the flexibility of adding new and diverse features. Experimental results on a set of focused What ...? and Which ...? questions show that our learned preference ranking methods perform better than alternative solutions to the task of answer typing. A gain of almost 0.2 in MRR for both the first appropriate and first correct answers is observed along with an increase in precision over the entire range of recall.

1 Introduction

Question answering (QA) systems have received a great deal of attention because they provide both a natural means of querying via questions and because they return short, concise answers. These two advantages simplify the task of finding information relevant to a topic of interest. Questions convey more than simply a natural language query; an implicit expectation of answer type is provided along with the question words. The discovery and exploitation of this implicit expected type is called *answer typing*.

We introduce an answer typing method that is sufficiently flexible to use a wide variety of features while at the same time providing a high level of performance. Our answer typing method avoids the use of pre-determined classes that are often lacking for unanticipated answer types. Because answer typing is only part of the QA task, a flexible answer typing model ensures that answer typing can be easily and usefully incorporated into a complete QA system. A discriminative preference ranking model with a preference for appropriate answers is trained and applied to unseen questions. In terms of Mean Reciprocal Rank (MRR), we observe improvements over existing systems of around 0.2 both in terms of the correct answer and in terms of appropriate responses. This increase in MRR brings the performance of our model to near the level of a full QA system on a subset of questions, despite the fact that we rely on answer typing features alone.

The amount of information given about the expected answer can vary by question. If the question contains a *question focus*, which we define to be the head noun following the *wh*-word such as *city* in "What *city* hosted the 1988 Winter Olympics?", some of the typing information is explicitly stated. In this instance, the answer is required to be a city. However, there is often additional information available about the type. In our example, the answer must plausibly host a Winter Olympic Games. The focus, along with the additional information, give strong clues about what are appropriate as responses.

We define an *appropriate* candidate answer as one that a user, who does not necessarily know the correct answer, would identify as a plausible answer to a given question. For most questions, there exist plausible responses that are not correct answers to the question. For our above question, the city of Vancouver is plausible even though it is not correct. For the purposes of this paper, we assume correct answers are a subset of appropriate candidates. Because answer typing is only intended to be a component of a full QA system, we rely on other components to help establish the true correctness of a candidate answer.

The remainder of the paper is organized as follows. Section 2 presents the application of discriminative preference rank learning to answer typing. Section 3 introduces the models we use

Proceedings of the 12th Conference of the European Chapter of the ACL, pages 666–674,
Athens, Greece, 30 March – 3 April 2009.

for learning appropriate answer preferences. Sections 4 and 5 discuss our experiments and their results, respectively. Section 6 presents prior work on answer typing and the use of discriminative methods in QA. Finally, concluding remarks and ideas for future work are presented in Section 7.

2 Preference Ranking

Preference ranking naturally lends itself to any problem in which the relative ordering between examples is more important than labels or values assigned to those examples. The classic example application of preference ranking (Joachims, 2002) is that of information retrieval results ranking. Generally, information retrieval results are presented in some ordering such that those higher on the list are either more relevant to the query or would be of greater interest to the user.

In a preference ranking task we have a set of candidates $c_1, c_2, ..., c_n$, and a ranking r such that the relation $c_i <_r c_j$ holds if and only if candidate c_i should be ranked higher than c_j, for $1 \leq i, j \leq n$ and $i \neq j$. The ranking r can form a total ordering, as in information retrieval, or a partial ordering in which we have both $c_i \not<_r c_j$ and $c_j \not<_r c_i$. Partial orderings are useful for our task of answer typing because they can be used to specify candidates that are of an equivalent rank.

Given some $c_i <_r c_j$, preference ranking only considers the difference between the feature representations of c_i and c_j ($\Phi(c_i)$ and $\Phi(c_j)$, respectively) as evidence. We want to learn some weight vector $\vec{w}$ such that $\vec{w} \cdot \Phi(c_i) > \vec{w} \cdot \Phi(c_j)$ holds for all pairs c_i and c_j that have the relation $c_i <_r c_j$. In other words, we want $\vec{w} \cdot (\Phi(c_i) - \Phi(c_j)) > 0$ and we can use some margin in the place of 0. In the context of Support Vector Machines (Joachims, 2002), we are trying to minimize the function:

$$V(\vec{w}, \vec{\xi}) = \frac{1}{2}\vec{w} \cdot \vec{w} + C \sum \xi_{i,j} \qquad (1)$$

subject to the constraints:

$$\forall (c_i <_r c_j) : \vec{w} \cdot (\Phi(c_i) - \Phi(c_j)) \geq 1 - \xi_{i,j} \quad (2)$$

$$\forall i, j : \xi_{i,j} \geq 0 \quad (3)$$

The margin incorporates slack variables $\xi_{i,j}$ for problems that are not linearly separable. This ranking task is analogous to the SVM classification task on the pairwise difference vectors $(\Phi(c_i) - \Phi(c_j))$, known as *rank constraints*. Unlike classification, no explicit negative evidence is required as $\vec{w} \cdot (\Phi(c_i) - \Phi(c_j)) = (-1)\vec{w} \cdot (\Phi(c_j) - \Phi(c_i))$. It is also important to note that no rank constraints are generated for candidates for which no order relation exists under the ranking r.

Support Vector Machines (SVMs) have previously been used for preference ranking in the context of information retrieval (Joachims, 2002). We adopt the same framework for answer typing by preference ranking. The SVM^{light} package (Joachims, 1999) implements the preference ranking of Joachims (2002) and is used here for learning answer types.

2.1 Application to Answer Typing

Assigning meaningful scores for answer typing is a difficult task. For example, given the question "What city hosted the 1988 Winter Olympics?" and the candidates New York, Calgary, and the word blue, how can we identify New York and Calgary as appropriate and the word blue as inappropriate? Scoring answer candidates is complicated by the fact that a gold standard for appropriateness scores does not exist. Therefore, we have no *a priori* notion that New York is better than the word blue by some amount v. Because of this, we approach the problem of answer typing as one of preference ranking in which the relative appropriateness is more important than the absolute scores.

Preference ranking stands in contrast to *classification*, in which a candidate is classified as appropriate or inappropriate depending on the values in its feature representation. Unfortunately, simple classification does not work well in the face of a large imbalance in positive and negative examples. In answer typing we typically have far more inappropriate candidates than appropriate candidates, and this is especially true for the experiments described in Section 4. This is indeed a problem for our system, as neither re-weighting nor attempting to balance the set of examples with the use of random negative examples were shown to give better performance on development data. This is not to say that *some* means of balancing the data would not provide comparable or superior performance, but rather that such a weighting or sampling scheme is not obvious.

An additional benefit of preference ranking over classification is that preference ranking models the *better-than* relationship between candidates. Typically a set of candidate answers are all related to a question in some way, and we wish to know which

of the candidates are better than others. In contrast, binary classification simply deals with the *is/is-not* relationship and will have difficulty when two responses with similar feature values are classified differently. With preference ranking, violations of some rank constraints will affect the resulting order of candidates, but sufficient ordering information may still be present to correctly identify appropriate candidates.

To apply preference ranking to answer typing, we learn a model over a set of questions $q_1, ..., q_n$. Each question q_i has a list of appropriate candidate answers $a_{(i,1)}, ..., a_{(i,u)}$ and a list of inappropriate candidate answers $b_{(i,1)}, ..., b_{(i,v)}$. The partial ordering r is simply the set

$$\forall i, j, k : \{a_{(i,j)} <_r b_{(i,k)}\} \quad (4)$$

This means that rank constraints are only generated for candidate answers $a_{(i,j)}$ and $b_{(i,k)}$ for question q_i and not between candidates $a_{(i,j)}$ and $b_{(l,k)}$ where $i \neq l$. For example, the candidate answers for the question "What city hosted the 1988 Winter Olympics?" are not compared with those for "What colour is the sky?" because our partial ordering r does not attempt to rank candidates for one question in relation to candidates for another. Moreover, no rank constraints are generated between $a_{(i,j)}$ and $a_{(i,k)}$ nor $b_{(i,j)}$ and $b_{(i,k)}$ because the partial ordering does not include orderings between two candidates of the same class. Given two appropriate candidates to the question "What city hosted the 1988 Winter Olympics?", New York and Calgary, rank constraints will not be created for the pair (New York, Calgary).

3 Methods

We begin with the work of Pinchak and Lin (2006) in which *question contexts* (dependency tree paths involving the *wh*-word) are extracted from the question and matched against those found in a corpus of text. The basic idea is that words that are appropriate as answers will appear in place of the *wh*-word in these contexts when found in the corpus. For example, the question "What city hosted the 1988 Winter Olympics?" will have as one of the question contexts "X hosted Olympics." We then consult a corpus to discover what replacements for X were actually mentioned and smooth the resulting distribution.

We use the model of Pinchak and Lin (2006) to produce features for our discriminative model.

Table 1: Feature templates

Pattern	Description
$E(t, c)$	Estimated count of term t in context c
$C(t, c)$	Observed count of term t in context c
$\sum_{t'} C(t', c)$	Count of all terms appearing in context c
$\sum_{c'} C(t, c')$	Count of term t in all contexts
$S(t)$	Count of the times t occurs in the candidate list

These features are mostly based on question contexts, and are briefly summarized in Table 1. Following Pinchak and Lin (2006), all of our features are derived from a limited corpus (AQUAINT); large-scale text resources are not required for our model to perform well. By restricting ourselves to relatively small corpora, we believe that our approach will easily transfer to other domains or languages (provided parsing resources are available).

To address the sparseness of question contexts, we remove lexical elements from question context paths. This removal is performed after feature values are obtained for the fully lexicalized path; the removal of lexical elements simply allows many similar paths to share a single learned weight. For example, the term Calgary in context $X \leftarrow$ *subject* $\leftarrow$ host $\rightarrow$ *object* $\rightarrow$ Olympics (X hosted Olympics) is used to obtain a feature value v that is assigned to a feature such as $C(\text{Calgary}, X \leftarrow subject \leftarrow * \rightarrow object \rightarrow *) = v$. Removal of lexical elements results in a space of 73 possible question contexts. To facilitate learning, all counts are log values and feature vectors are normalized to unit length.

The estimated count of term t in context c, $E(t, c)$, is a component of the model of Pinchak and Lin (2006) and is calculated according to:

$$E(t, c) = \sum_{\chi} \Pr(\chi|t) C(\chi, c) \quad (5)$$

Essentially, this equation computes an expected count for term t in question c by observing how likely t is to be part of a cluster χ ($\Pr(\chi|t)$) and then observing how often terms of cluster χ occur in context c ($C(\chi, c)$). Although the model of Pinchak and Lin (2006) is significantly more

complex, we use their core idea of cluster-based smoothing to decide how often a term t will occur in a context c, regardless of whether or not t was actually observed in c within our corpus. The Pinchak and Lin (2006) system is unable to assign individual weights to different question contexts, even though not all question contexts are equally important. For example, the Pinchak and Lin (2006) model is forced to consider a question focus context (such as "*X* is a city") to be of equal importance to non-focus contexts (such as "*X* host Olympics"). However, we have observed that it is more important that candidate *X* is a city than it hosted an Olympics in this instance. Appropriate answers are required to be cities even though not all cities have hosted Olympics. We wish to address this problem with the use of discriminative methods.

The observed count features of term t in context c, $C(t, c)$, are included to allow for combination with the estimated values from the model of Pinchak and Lin (2006). Because Pinchak and Lin (2006) make use of cluster-based smoothing, errors may occur. By including the observed counts of term t in context c, we hope to allow for the use of more accurate statistics whenever they are available, and for the smoothed counts in cases for which they are not.

Finally, we include the frequency of a term t in the list of candidates, $S(t)$. The idea here is that the correct and/or appropriate answers are likely to be repeated many times in a list of candidate answers. Terms that are strongly associated with the question and appear often in results are likely to be what the question is looking for.

Both the $C(t, c)$ and $S(t)$ features are extensions to the Pinchak and Lin (2006) model and can be incorporated into the Pinchak and Lin (2006) model with varying degrees of difficulty. The value of $S(t)$ in particular is highly dependent on the means used to obtain the candidate list, and the distribution of words over the candidate list is often very different from the distribution of words in the corpus. Because this feature value comes from a different source than our other features, it would be difficult to use in a non-discriminative model.

Correct answers to our set of questions are obtained from the TREC 2002-2006 results (Voorhees, 2002). For appropriateness labels we turn to human annotators. Two annotators were instructed to label a candidate as appropriate if that candidate was believable as an answer, even if that candidate was not correct. For a question such as "What city hosted the 1988 Winter Olympics?", all cities should be labeled as appropriate even though only Calgary is correct. This task comes with a moderate degree of difficulty, especially when dealing with questions for which appropriate answers are less obvious (such as "What kind of a community is a Kibbutz?"). We observed an inter-annotator (kappa) agreement of 0.73, which indicates substantial agreement. This value of kappa conveys the difficulty that even human annotators have when trying to decide which candidates are appropriate for a given question. Because of this value of kappa, we adopt strict gold standard appropriateness labels that are the intersection of the two annotators' labels (i.e., a candidate is only appropriate if both annotators label it as such, otherwise it is inappropriate).

We introduce four different models for the ranking of appropriate answers, each of which makes use of appropriateness labels in different ways:

Correctness Model: Although appropriateness and correctness are not equivalent, this model deals with distinguishing correct from incorrect candidates in the hopes that the resulting model will be able to perform well on finding both correct and appropriate answers. For learning, correct answers are placed at a rank above that of incorrect candidates, regardless of whether or not those candidates are appropriate. This represents the strictest definition of appropriateness and requires no human annotation.

Appropriateness Model: The correctness model assumes only correct answers are appropriate. In reality, this is seldom the case. For example, documents or snipppets returned for the question "What country did Catherine the Great rule?" will contain not only Russia (the correct answer), but also Germany (the nationality of her parents) and Poland (her modern-day birthplace). To better address this overly strict definition of appropriateness, we rank all candidates labeled as appropriate above those labeled as inappropriate, without regards to correctness. Because we want to learn a model for appropriateness, training on appropriateness rather than correctness information should produce a model closer to what we desire.

Combined Model: Discriminative preference ranking is not limited to only two ranks. We combine the ideas of correctness and appropri-

ateness together to form a three-rank combined model. This model places correct answers above appropriate-but-incorrect candidates, which are in turn placed above inappropriate-and-incorrect candidates.

Reduced Model: Both the appropriateness model and the combined model incorporate a large number of rank constraints. We can reduce the number of rank constraints generated by simply removing all appropriate, but incorrect, candidates from consideration and otherwise following the correctness model. The main difference is that some appropriate candidates are no longer assigned a low rank. By removing appropriate, but incorrect, candidates from the generation of rank constraints, we no longer rank correct answers above appropriate candidates.

4 Experiments

To compare with the prior approach of Pinchak and Lin (2006), we use a set of *what* and *which* questions with question focus (questions with a noun phrase following the *wh*-word). These are a subset of the more general *what*, *which*, and *who* questions dealt with by Pinchak and Lin (2006). Although our model can accommodate a wide range of *what*, *which*, *when*, and *who* questions, the focused *what* and *which* questions are an easily identifiable subclass that are rarely definitional or otherwise complex in terms of the desired answer. We take the set of focused *what* and *which* questions from TREC 2002-2006 (Voorhees, 2002) comprising a total of 385 questions and performed 9-fold cross-validation, with one dedicated development partition (the tenth partition). The development partition was used to tune the regularization parameter of the SVM used for testing.

Candidates are obtained by submitting the question as-is to the Google search engine and chunking the top 20 snippets returned, resulting in an average of 140 candidates per question. Google snippets create a better confusion set than simply random words for appropriate and inappropriate candidates; many of the terms found in Google snippets are related in some way to the question. To ensure a correct answer is present (where possible), we append the list of correct answers to the list of candidates.

As a measure of performance, we adopt Mean Reciprocal Rank (MRR) for both correct and appropriate answers, as well as precision-recall for appropriate answers. MRR is useful as a measure of overall QA system performance (Voorhees, 2002), but is based only on the top correct or appropriate answer encountered in a ranked list. For this reason, we also show the precision-recall curve to better understand how our models perform.

We compare our models with three alternative approaches, the simplest of which is *random*. For random, the candidate answers are randomly shuffled and performance is averaged over a number of runs (100). The *snippet frequency* approach orders candidates based on their frequency of occurrence in the Google snippets, and is simply the $S(t)$ feature of our discriminative models in isolation. We remove terms comprised solely of question words from all approaches to prevent question words (which tend to be very frequent in the snippets) from being selected as answers. The last of our alternative systems is an implementation of the work of Pinchak and Lin (2006) in which the output probabilities of their model are used to rank candidates.

4.1 Results

Figures 1 and 2 show the MRR results and precision-recall curve of our correctness model against the alternative approaches. In comparison to these alternative systems, we show two versions of our correctness model. The first uses a linear kernel and is able to outperform the alternative approaches. The second uses a radial basis function (RBF) kernel and exhibits performance superior to that of the linear kernel. This suggests a degree of non-linearity present in the data that cannot be captured by the linear kernel alone. Both the training and running times of the RBF kernel are considerably larger than that of the linear kernel. The accuracy gain of the RBF kernel must therefore be weighed against the increased time required to use the model.

Figures 3 and 4 give the MRR results and precision-recall curves for our additional models in comparison with that of the correctness model. Although losses in MRR and precision are observed for both the appropriate and combined model using the RBF kernel, the linear kernel versions of these models show slight performance gains.

Figure 1: MRR results for the correctness model

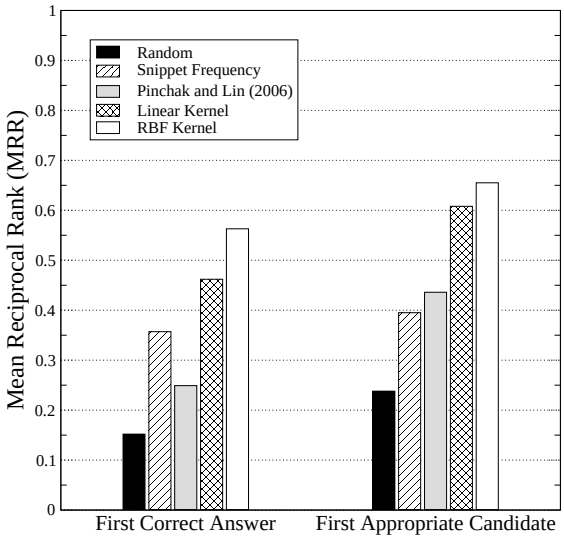
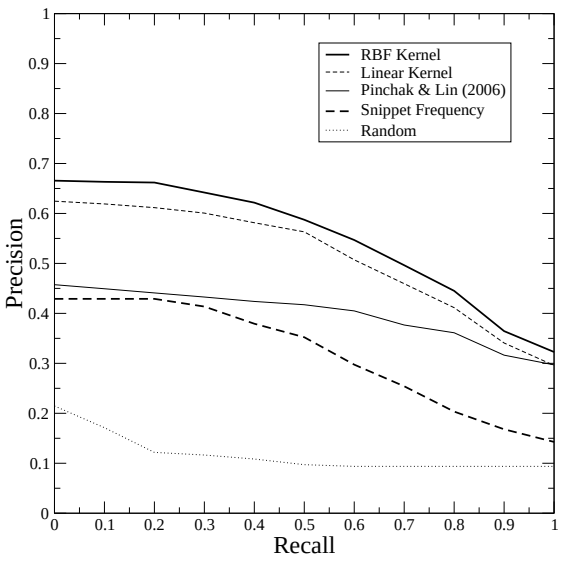


Figure 2: Precision-recall of appropriate candidates under the correctness model



5 Discussion of Results

The results of our correctness model, found in Figures 1 and 2 show considerable gains over our alternative systems, including that of Pinchak and Lin (2006). The Pinchak and Lin (2006) system was specifically designed with answer typing in mind, although it makes use of a brittle generative model that does not account for ranking of answer candidates nor for the variable strength of various question contexts. These results show that our discriminative preference ranking approach creates a better model of both correctness and appropriateness via weighting of contexts, preference rank learning, and with the incorporation of additional related features (Table 1). The last feature, snippet frequency, is not particularly strong on its own, but can be easily incorporated into our discriminative model. The ability to add a wide variety of potentially helpful features is one of the strengths of discriminative techniques in general.

By moving away from simply correct answers in the correctness model and incorporating labeled appropriate examples in various ways, we are able to further improve upon the performance of our approach. Training on appropriateness labels instead of correct answers results in a loss in MRR for the first correct answer, but a gain in MRR for the first appropriate candidate. Unfortunately, this does not carry over to the entire range of precision over recall. For the linear kernel, our three additional models (appropriateness, combined, and reduced) show consistent improvements over the correctness model, but with the RBF kernel only the reduced model produces a meaningful change.

The precision-recall curves of Figures 2 and 4 show remarkable consistency across the full range of recall, despite the fact that candidates exist for which feature values cannot easily be obtained. Due to tagging and chunking errors, ill-formed candidates may exist that are judged appropriate by the annotators. For example, “explorer Hernando Soto” is a candidate marked appropriate by both annotators to the question “What Spanish explorer discovered the Mississippi River?” However, our context database does not include the phrase “explorer Hernando Soto” meaning that only a few features will have non-zero values. Despite these occasional problems, our models are able to rank most correct and appropriate candidates high in a ranked list.

Finally, we examine the effects of training set size on MRR. The learning curve for a single partitioning under the correctness model is presented in Figure 5. Although the model trained with the RBF kernel exhibits some degree of instability below 100 training questions, both the linear and RBF models gain little benefit from additional training questions beyond 100. This may be due to the fact that the most common unlexicalized question contexts have been observed in the first

Figure 3: MRR results (RBF kernel)

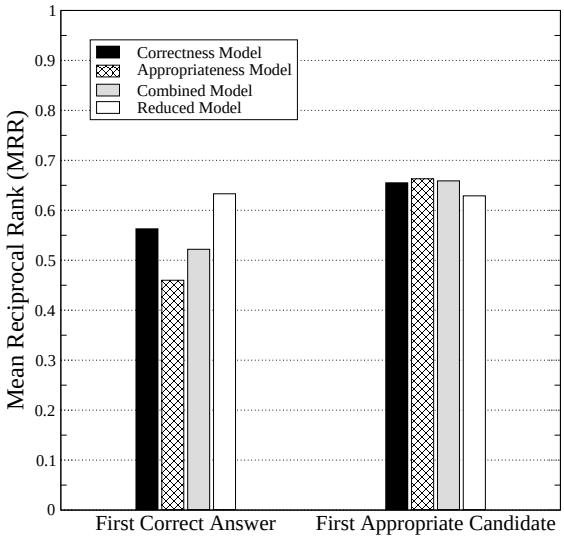
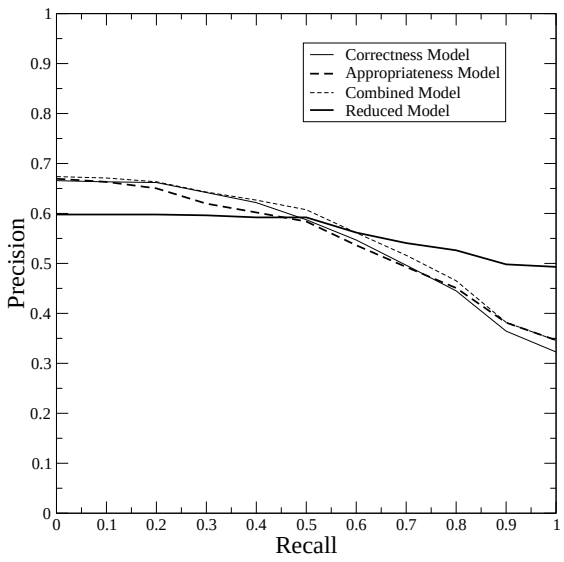


Figure 4: Precision-recall of appropriate (RBF kernel)



100 training examples and so therefore additional questions simply repeat the same information. Requiring only a relatively small number of training examples means that an effective model can be learned with relatively little input in the form of question-answer pairs or annotated candidate lists.

6 Prior Work

The expected answer type can be captured in a number of possible ways. By far the most common is the assignment of one or more predefined types to a question during a question analysis phase. Although the vast majority of the approaches to answer type detection make use of rules (either partly or wholly) (Harabagiu et al., 2005; Sun et al., 2005; Wu et al., 2005; Mollá and Gardiner, 2004), a few notable learned methods for answer type detection exist.

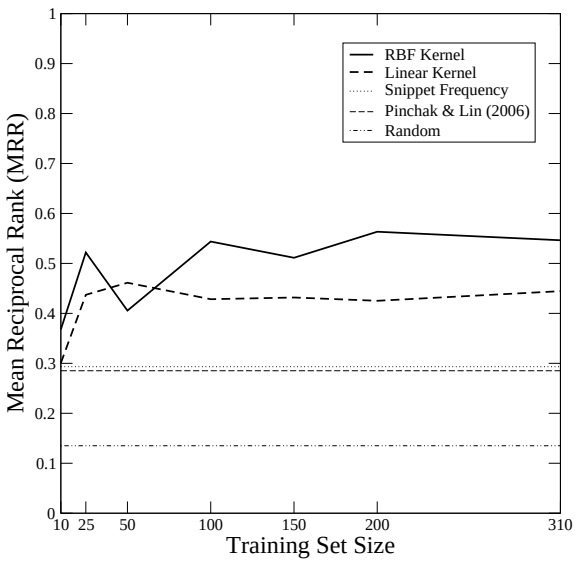
One of the first attempts at learning a model for answer type detection was made by Ittycheriah et al. (2000; 2001) who learn a maximum entropy classifier over the Message Understanding Conference (MUC) types. Those same MUC types are then assigned by a named-entity tagger to identify appropriate candidate answers. Because of the potential for unanticipated types, Ittycheriah et al. (2000; 2001) include a *Phrase* type as a catch-all class that is used when no other class is appropriate. Although the classifier and named-entity tagger are shown to be among the components with the lowest error rate in their QA system, it is not clear how much benefit is obtained from using a relatively coarse-grained set of classes.

The approach of Li and Roth (2002) is similar in that it uses learning for answer type detection. They make use of multi-class learning with a Sparse Network of Winnows (SNoW) and a two-layer class hierarchy comprising a total of fifty possible answer types. These finer-grained classes are of more use when computing a notion of appropriateness, although one major drawback is that no entity tagger is discussed that can identify these types in text. Li and Roth (2002) also rely on a rigid set of classes and so run the risk of encountering a new question of an unseen type.

Pinchak and Lin (2006) present an alternative in which the probability of a term being appropriate to a question is computed directly. Instead of assigning an answer type to a question, the question is broken down into a number of possibly overlapping contexts. A candidate is then evaluated as to how likely it is to appear in these contexts. Unfortunately, Pinchak and Lin (2006) use a brittle generative model when combining question contexts that assumes all contexts are equally important. This assumption was dealt with by Pinchak and Lin (2006) by discarding all non-focus contexts with a focus context is present, but this is not an ideal solution.

Learning methods are abundant in QA research

Figure 5: Learning curve for MRR of the first correct answer under the correctness model



and have been applied in a number of different ways. Ittycheriah et al. (2000) created an entire QA system based on maximum entropy components in addition to the question classifier discussed above. Ittycheriah et al. (2000) were able to obtain reasonable performance from learned components alone, although future versions of the system use non-learned components in addition to learned components (Prager et al., 2003). The JAVELIN I system (Nyberg et al., 2005) uses a SVM during the answer/information extraction phase. Although learning is applied in many QA tasks, very few QA systems rely solely on learning. Compositional approaches, in which multiple distinct QA techniques are combined, also show promise for improving QA performance. Echihabi et al. (2003) use three separate answer extraction agents and combine the output scores with a maximum entropy re-ranker.

Surdeanu et al. (2008) explore preference ranking for advice or "how to" questions in which a unique correct answer is preferred over all other candidates. Their focus is on complex-answer questions in addition to the use of a collection of user-generated answers rather than answer typing. However, their use of preference ranking mirrors the techniques we describe here in which the relative difference between two candidates at different ranks is more important than the individual candidates.

7 Conclusions and Future Work

We have introduced a means of flexible answer typing with discriminative preference rank learning. Although answer typing does not represent a complete QA system, it is an important component to ensure that those candidates selected as answers are indeed appropriate to the question being asked. By casting the problem of evaluating appropriateness as one of preference ranking, we allow for the learning of what differentiates an appropriate candidate from an inappropriate one.

Experimental results on focused *what* and *which* questions show that a discriminatively trained preference rank model is able to outperform alternative approaches designed for the same task. This increase in performance comes from both the flexibility to easily combine a number of weighted features and because comparisons only need to be made between appropriate and inappropriate candidates. A preference ranking model can be trained from a relatively small set of example questions, meaning that only a small number of question/answer pairs or annotated candidate lists are required.

The power of an answer typing system lies in its ability to identify, in terms of some given query, appropriate candidates. Applying the flexible model described here to a domain other than question answering could allow for a more focused set of results. One straight-forward application is to apply our model to the process of information or document retrieval itself. Ensuring that there are terms present in the document appropriate to the query could allow for the intelligent expansion of the query. In a related vein, queries are occasionally comprised of natural language text fragments that can be treated similarly to questions. Rarely are users searching for simple mentions of the query in pages; we wish to provide them with something more useful. Our model achieves the goal of finding those appropriate related concepts.

Acknowledgments

We would like to thank Debra Shiau for her assistance annotating training and test data and the anonymous reviewers for their insightful comments. We would also like to thank the Alberta Informatics Circle of Research Excellence and the Alberta Ingenuity Fund for their support in developing this work.

References

A. Echihabi, U. Hermjakob, E. Hovy, D. Marcu, E. Melz, and D. Ravichandran. 2003. Multiple-Engine Question Answering in TextMap. In *Proceedings of the Twelfth Text REtrieval Conference (TREC-2003)*, Gaithersburg, Maryland.

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 (TREC-2005)*, Gaithersburg, Maryland.

A. Ittycheriah, M. Franz, W-J. Zhu, A. Ratnaparkhi, and R. Mammone. 2000. IBM's Statistical Question Answering System. In *Proceedings of the 9th Text REtrieval Conference (TREC-9)*, Gaithersburg, Maryland.

A. Ittycheriah, M. Franz, and S. Roukos. 2001. IBM's Statistical Question Answering System – TREC-10. In *Proceedings of the 10th Text REtrieval Conference (TREC-10)*, Gaithersburg, Maryland.

T. Joachims. 1999. Making Large-Scale SVM Learning Practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT-Press.

T. Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM.

X. Li and D. Roth. 2002. Learning Question Classifiers. In *Proceedings of the International Conference on Computational Linguistics (COLING 2002)*, pages 556–562.

D. Mollá and M. Gardiner. 2004. AnswerFinder - Question Answering by Combining Lexical, Syntactic and Semantic Information. In *Proceedings of the Australian Language Technology Workshop (ALTW 2004*, pages 9–16, Sydney, December.

E. Nyberg, R. Frederking, T. Mitamura, M. Bilotti, K. Hannan, L. Hiyakumoto, J. Ko, F. Lin, L. Lita, V. Pedro, and A. Schlaikjer. 2005. JAVELIN I and II Systems at TREC 2005. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC-2005)*, Gaithersburg, Maryland.

C. Pinchak and D. Lin. 2006. A Probabilistic Answer Type Model. In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006)*, Trento, Italy, April.

J. Prager, J. Chu-Carroll, K. Czuba, C. Welty, A. Ittycheriah, and R. Mahindru. 2003. IBM's PIQUANT in TREC2003. In *Proceedings of the Twelfth Text REtrieval Conference (TREC-2003)*, Gaithersburg, Maryland.

R. Sun, J. Jiang, Y.F. Tan, H. Cui, T-S. Chua, and M-Y. Kan. 2005. Using Syntactic and Semantic Relation Analysis in Question Answering. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC-2005)*, Gaithersburg, Maryland.

M. Surdeanu, M. Ciaramita, and H. Zaragoza. 2008. Learning to rank answers on large online QA collections. In *Proceedings of the 46th Annual Meeting for the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT)*, pages 719–727, Columbus, Ohio, June. Association for Computational Linguistics.

E.M. Voorhees. 2002. Overview of the TREC 2002 Question Answering Track. In *Proceedings of TREC 2002*, Gaithersburg, Maryland.

M. Wu, M. Duan, S. Shaikh, S. Small, and T. Strzalkowski. 2005. ILQUA – An IE-Driven Question Answering System. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC-2005)*, Gaithersburg, Maryland.