Answer Typing for Information Retrieval

Christopher Pinchak
Dept. of Computing Science
Univesity of Alberta
Edmonton, Alberta, Canada
pinchak@cs.ualberta.ca

Davood Rafiei
Dept. of Computing Science
Univesity of Alberta
Edmonton, Alberta, Canada
drafiei@cs.ualberta.ca

Dekang Lin
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, California
lindek@google.com

ABSTRACT

Answer typing is commonly thought of as finding appropriate responses to given questions. We extend the notion of answer typing to information retrieval to ensure results contain plausible answers to queries. Identification of a large class of applicable queries is performed using a discriminative classifier, and discriminative preference ranking methods are employed for the selection of type-appropriate terms. Experimental results show that type-appropriate terms identified by the model are superior to terms most commonly associated with the query, providing strong evidence that answer typing techniques can find meaningful and appropriate terms. Further experiments show that snippets containing correct answers are ranked higher by our model than by the baseline Google search engine in those instances in which a query does indeed seek a short answer.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.7 [**Artificial Intelligence**]: Natural Language Processing

General Terms

Algorithms, Experimentation, Measurement, Performance

1. INTRODUCTION

The tasks of question answering (QA) and information retrieval (IR) both provide a means of locating information relevant to a request, albeit in slightly different ways. QA provides a natural question and answer interface that reduces the amount of information returned, but requires additional work on the part of the system to find correct answers. This additional work has led to QA systems being deployed on a much smaller scale than IR systems. IR systems are often only aware of questions in a limited sense, such as answering only questions matching a certain pattern, and exhibit reduced performance on other questions.

CIKM'09, November 2–6, 2009, Hong Kong, China.

In this paper, we seek to apply the QA notion of answer typing to generate better responses to queries. Specifically, we wish to identify type-appropriate terms for these queries such that standard IR responses (such as snippets) include terms that satisfy the user's information needs. To take advantage of typeability we introduce methods to 1) identify queries that can benefit from typing, 2) identify type-appropriate terms for these queries, and 3) use these terms to promote results.

Experimental results show that snippet terms scored by our model are preferred nearly twice as often as the most frequent terms shared by the top 100 snippets. Further experiments show that snippets containing answers are ranked higher by our model than by the Google baseline for those cases in which queries have a complete or partial short answer. An increase of 0.05 in mean reciprocal rank (MRR) is observed, resulting in a MRR value of of 0.567. This MRR value is quite high, especially considering the fact that multiple results are displayed to a judge.

Information retrieval is a central component of many QA systems [5, 15]. Although IR techniques have been investigated with the specific goal of QA in mind [14], the application of QA techniques specifically with IR in mind has been largely unexplored. However, because document retrieval (the IR component of QA) can benefit from increased awareness of the question [23, 2], certain aspects of the QA process have been transplanted into document retrieval. Although the application of Natural Language Processing (NLP) techniques to the IR task is not a new concept [22, 9], ours is one of the few works that seeks to transfer the notion of an automatically-detected answer type into IR or document retrieval. However, work on templated queries [10] allows for the incorporation of type information specified by the user.

Answer typing is a central component for many QA systems (e.g., [3, 5]). In the context of QA, answer typing seeks to identify those candidate answers that are plausible or appropriate as responses to the question. Answer typing is most often performed by assigning one or more *answer type classes* to the question, either via rules or via *question classification* such as that performed by the system of Li and Roth [12]. The work we follow for answer typing in this paper is that of Pinchak et al. [20] in which answer typing is performed without the use of answer type classes.

2. IDENTIFYING QUERIES

Before we can apply answer typing to queries, we must first identify those queries that are naturally amenable to typing. A query stream, such as the one found in the AOL

Table 1: Identification feature templates

Pattern	Description
LHS_w	root noun to the left of the preposition
LHS_c	cluster of the root noun
RHS_w	preposition and the root of the prepositional attachment
RHS_c	preposition and cluster of the root of the prep. attachment
has_picture	query contains the word "picture" or "photo"
has_map	query contains the word "map"
initial_the	query starts with the word "the"
rearranged	find a non-prepositional wording of this query in the AOL data set

Table 2: Answer typing feature templates

Pattern	Description
$E(t, c)$	Expected 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 context c in the corpus
$\sum_{c'} C(t, c')$	Count of term t in the corpus
$W(t)$	Estimated depth of t in the WordNet hierarchy
$S(t)$	Count of the times t occurs in the candidate list
$LHS(t, q)$	Flag for when the LHS of query q is a substring of t
$U(t)$	Flag for when t contains capitalized letters
$T(t)$	Number of terms comprising candidate t

query logs [17], contains a wide variety of queries searching for different kinds of information. Queries can be roughly divided into *short-answer informational*, *long-answer informational*, *navigational*, and *definition*. Informational queries are seeking some piece of information about a subject, and the response may be short-answer (e.g., "cities in Canada") or long-answer (e.g., "causes of WWII"). In contrast to informational queries, navigational queries are seeking some specific page (e.g., "Google image search") and definition queries are looking for some comprehensive source of information on a subject (e.g., the Wikipedia page for Britney Spears).

Lapata and Keller [11] describe a task of compound noun interpretation in which some compound noun phrases, such as "war stories," can be rearranged or rewritten as prepositional queries, such as "stories about war." Many queries encountered in a query stream are compound noun phrases and so we wish to develop techniques that cover these queries. Therefore, we propose that compound noun queries can be *normalized* by transforming them into their most likely prepositional rewriting thus simplifying our task to considering only those queries that already contain prepositions.

Although the set of prepositional queries is homogeneous in structure, we must further filter the queries to remove those unlikely to benefit from a notion of answer type. For this task, we make use of large-margin discriminative learning as implemented by a Support Vector Machine (SVM) classifier [7] for which we identify the set of useful features summarized in Table 1. Because we only consider prepositional queries, we can break each query into a left-hand side (LHS) and right-hand side (RHS). The LHS contains only the root noun of the query. The RHS includes both the preposition and the root of the prepositional attachment. Cluster information is used to increase the amount of overlap between queries that are not identical.

The purpose of our SVM is to identify queries that benefit from our notion of typing. To provide training data, the first author of this paper labeled a set of 2000 queries according to whether or not the answer typing methods described next are likely to be of benefit to the query. A single annotator was used because of the high degree of subjectivity of the task; it is not always clear what exactly defines a typeable query and some familiarity with the answer typing method is required to identify cases where typing is likely to help. Experiments on a small test set of 200 queries show an accuracy of 87.5% for the query identification model (25 errors), with errors being divided according to the proportion of positive and negative examples (one-third and two-thirds, respectively). Our experiments in Section 5 show that this level of performance produces results comparable to our baseline approach, and assuming perfect accuracy leads to clear improvements over the baseline.

3. PREFERENCE RANKING

Answer typing for QA typically takes the form of assigning one or more pre-defined type categories to a given question [6, 12]. Pinchak and Lin [20] approach the problem of answer typing for QA by directly modeling the probability of a candidate answer being appropriate to a given question without the use of an intermediary type category. Following this work, we make use of discriminative preference ranking to order a given list of candidate answers according to how appropriate they are to a given question. Because Pinchak et al. [20] use support vector machines (SVMs), we use the SVM^{light} package [8] to perform discrimitive preference ranking.

4. TYPING IR RESULTS

We begin by considering only those prepositional queries identified as typeable by the model discussed above in Section 2. From the queries, we generate *query contexts* from the LHS and from the combination of LHS with RHS. For example, the query "cities in Canada" generates the context "X is a city" from the LHS and "X is a city in Canada" from the combination of LHS and RHS. Here X is a placeholder for the expected appropriate term.

Given that these queries were originally intended for information retrieval, we submit them as-is to the popular Google search engine and obtain a ranked list of results along with their snippets. The top 100 snippets are tagged [19] and chunked [18] to extract a set of candidates that are then ranked by the model. These ranked candidates form an ordered *candidate list*.

Table 3: Top 20 Terms SxS

Total queries	996
Our terms preferred	365 (37%)
Most frequent terms preferred	194 (19%)
Both good	83 (8%)
Both bad	108 (11%)
Indeterminate	246 (25%)

A discriminative preference ranking model grants us the flexibility of using many diverse features. The feature templates we use are in Table 2. As the basis of the model, we rely on an expected value of candidate answer t appearing in context c. This value is calculated using a list of similar words along with their similarity values. To balance these expected counts, we include the actual observed counts of candidate t in context c, $C(t, c)$, along with the individual counts of the candidate t and context c in our corpus [1].

To these basic counts, we add five additional kinds of features that do not rely on query contexts. The first, denoted by $W(t)$, is the estimated depth of the candidate t in the WordNet hierarchy [4]. Should the candidate not appear in WordNet, we estimate the depth of the candidate by averaging the depth of words with high similarity (t') according to the clusters of Pantel and Lin [16]. We include a feature for the number of times t occurs on the candidate list, $S(t)$. Through our extraction of candidates from snippets, appropriate candidates are often repeated a number of times. $LHS(t, q)$ is simply a flag that fires whenever the LHS of a query (such as "city" in "cities in Canada") appears as part of the candidate. We also include a flag that fires when a candidate contains one or more capitalized letters, $U(t)$, and the integer number of space-delineated words in the candidate, $T(t)$.

For training data, two annotators identified appropriate candidates in the top 20 snippets returned for each query from a total of 200 queries randomly selected from the positive training examples used for query identification (Section 2). We observed an inter-annotator agreement (kappa) of 0.68, which is relatively low. Because of this relatively low level of agreement, we chose to train on the intersection of the labels.

Once the candidates have been ordered according to our preference ranking model, we select the top 20 candidates for scoring snippets. The score of a snippet s is calculated as the average number of candidates per *fragment*:

$$score(s) = \frac{\sum_{t \in top20} in(t, s)}{frag(s)} \quad (1)$$

where $in(t, s) = 1$ if snippet s contains candidate t (0 otherwise) and $frag(s)$ is the number of fragments delineated by "..." in the snippet. Fewer fragments indicates a more cohesive snippet; highly-fragmented snippets require more appropriate candidates to receive a high score.

Given that Google often performs well at returning relevant results, we do not wish to venture too far from the Google ordering without good reason. To this end, we use an interpolated model most often employed for smoothing [13]. Our final score for a snippet is therefore:

$$inter(s) = \alpha \times MRR_{score(s)} + (1 - \alpha) \times MRR_{orig} \quad (2)$$

Table 4: Top-5 MRR

Ranking method	MRR
Original Google order	0.514
Reranked by our model	0.567

5. EXPERIMENTS

The primary goal of applying answer typing techniques to IR queries is to improve the relevance of results returned to the user. Given that results will be rescored based in part on the terms they contain, we propose two different experiments to evaluate the quality of this reordering. These experiments make use of an α parameter of 0.4, determined using a held-out development set. When looking at snippets, we examine only the top five snippets.

5.1 Candidate Ranking

Our candidate ranking experiment is meant to measure the quality of terms selected to rerank snippets in comparison with terms that are frequent. For this task, we introduce a *side-by-side (SxS)* experiment in which results from two alternatives are compared next to one another. Annotators are then asked to choose which side provides a better result. Displaying a list of appropriate terms does not conform to the task of IR, but allows us to determine whether or not our model can find relevant results.

Annotations for this experiment are collected from the Amazon Mechanical Turk (AMT) system.[1] AMT results, when averaged, have been shown to have high agreement with expert annotators [21] even though Turkers are not experts. We take advantage of this by requiring a minimum of five judgements on any one set of query responses. One side is preferred over the other if and only if we observe a majority of votes (i.e., ≥ 3) for that side.

Table 3 shows the results of comparing our terms with the most frequent terms for a set of 996 queries judged as typeable by the query identification model of Section 2. This set of queries includes queries erroneously identified as answerable because useful terms can exist for queries that are not answerable by short answers alone. The results in Table 3 show the clear advantage of our model over the most frequent terms indicating that our model is able to identify terms appropriate to this particular subset of queries.

5.2 Snippet Reranking

The encouraging results of the previous section lead to a further experiment in which snippets are reranked according to our model and compared with the original Google ordering. This experiment deals only with those queries for which there exists some short answer. Of the 996 queries used in the prior SxS experiment, 331 are strictly determined to be short-answerable. Snippets provide additional information and context for answers and are the expected response to IR queries. As a result, snippets that include a short answer to the query allow a user to find desired information along with some context without having to visit additional external Web pages.

Annotators were asked to identify the position of the first snippet containing a short answer to the query. Two annotators were used instead of Amazon Mechanical Turk due to

[1] `http://www.mturk.com/`

the attention to detail required to identify short answers in snippet text. The answer was allowed to be partial to cover cases in which a list of answers is sought or for which more than one answer is correct. Only the top 5 snippets of our two systems were presented to the annotators to produce a measure of top-5 reciprocal rank. The results for both systems are presented in Table 4. The two annotators agree on over 80% of the queries, indicating high confidence in these values.

The results of Table 4 show a slight improvement over the high performance of Google-ranked snippets. Given the already high performance of Google, we conclude that Google offers relatively few opportunities for which we can improve results. The fact that we show a significant improvement for this set of queries means that we are able to capitalize on those rare situations in which Google provides an overly general response.

Given the fact that Google performs well at finding correct answers for our set of queries, it is worthwhile to examine how often our model has an opportunity to improve performance. We observed that 176 (53%) of queries are answered by the first or second snippet provided by Google. This subset of queries offers only a slight opportunity for improvement. In spite of this fact, we observe a significant improvement in the number of queries correctly answered by the first two snippets provided by our interpolated model, up from 176 (53%) to 217 (66%). This means that a correct answer exists in the first two snippets for 2/3 of the queries that are identified as strictly answerable. Placing such snippets high in a ranked list is very important for an IR system, and our model successfully increases the number of queries that can be answered by considering only the first two snippets provided by our model.

6. CONCLUSIONS

Incorporating a notion of type-awareness into information retrieval is desirable for those queries that can benefit from typing. We have presented here a simple method by which such queries can be identified along with a means of identifying terms that are type-appropriate for a query. On their own, these terms are much better than common terms found in results; if a user is searching for one or a few specific instances then these terms may well satisfy their information needs. When these terms are used to rerank snippets, we observe a slight but significant improvement in finding correct answers to queries.

7. REFERENCES

[1] The AQUAINT Corpus of English News Text. Linguistic Data Corporation, 2002.

[2] M. Bilotti, P. Ogilvie, J. Callan, and E. Nyberg. Structured Retrieval for Question Answering. In *Proceedings of SIGIR 2007*, 2007.

[3] E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. Data-Intensive Question Answering. In *Proceedings of TREC 2001*, Gaithersburg, Maryland, 2001.

[4] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

[5] S. Harabagiu, D. Moldovan, C. Clark, M. Bowden, A. Hickl, and P. Wang. Employing Two Question Answering Systems in TREC-2005. In *Proceedings of TREC-2005*, 2005.

[6] A. Ittycheriah, M. Franz, and S. Roukos. IBM's Statistical Question Answering System – TREC-10. In *Proceedings of TREC-10*, 2001.

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

[8] T. Joachims. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of KDD-2002*. ACM, 2002.

[9] B. Katz and J. Lin. Selectively Using Relations to Improve Precision in Question Answering. In *Proceedings of EACL 2003 Workshop on Natural Language Processing for Question Answering*, pages 43–50, 2003.

[10] G. Kumaran and J. Allan. Information Retrieval Techniques for Templated Queries. In *Proceedings of RIAO 2007*, 2007.

[11] M. Lapata and F. Keller. Web-based Models for Natural Language Processing. *ACM Transactions on Speech and Language Processing*, 2(1):1–30, Feb. 2005.

[12] X. Li and D. Roth. Learning Question Classifiers. In *Proceedings of COLING 2002*, pages 556–562, 2002.

[13] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[14] C. Monz. *From Document Retrieval to Question Answering*. PhD thesis, University of Amsterdam, 2003.

[15] E. Nyberg, R. Frederking, T. Mitamura, M. Bilotti, K. Hannan, L. Hiyakumoto, J. Ko, F. Lin, L. Lita, V. Pedro, and A. Schlaikjer. JAVELIN I and II Systems at TREC 2005. In *Proceedings of TREC-2005*, 2005.

[16] P. Pantel and D. Lin. Document Clustering with Committees. In *Proceedings of SIGIR 2002*, pages 199–206, 2002.

[17] G. Pass, A. Chowdhury, and C. Torgeson. A Picture of Search. In *Proceedings of the First International Conference on Scalable Information Systems*, 2006.

[18] X. Phan. CRFChunker: CRF English Phrase Chunker. http://crfchunker.sourceforge.net/, 2006.

[19] X. Phan. CRFTagger: CRF English POS Tagger. http://crftagger.sourceforge.net/, 2006.

[20] C. Pinchak, D. Lin, and D. Rafiei. Flexible Answer Typing with Discriminative Preference Ranking. In *Proceedings of EACL 2009*, pages 666 – 674, 2009.

[21] R. Snow, B. O'Connor, D. Jurafsky, and A. Ng. Cheap and Fast – But is it Good? Evalating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the EMNLP 2008*, pages 254–263, 2008.

[22] T. Strzalkowski, L. Guthrie, J. Karlgren, J. Leistensnider, F. Lin, J. Pérez-Carballo, T. Straszheim, J. Wang, and J. Wilding. Natural Language Information Retrieval: TREC-5 Report. In *Proceedings of TREC-5*, 1996.

[23] J. Tiedemann. Improving Passage Retrieval in Question Answering using NLP. In *Proceedings of the 12th Portuguese Conference on Artificial Intelligence (EPIA)*, 2005.