A Probabilistic Answer Type Model

Christopher Pinchak
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada
`pinchak@cs.ualberta.ca`

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

Abstract

All questions are implicitly associated with an expected answer type. Unlike previous approaches that require a predefined set of question types, we present a method for dynamically constructing a probability-based answer type model for each different question. Our model evaluates the appropriateness of a potential answer by the probability that it fits into the question contexts. Evaluation is performed against manual and semi-automatic methods using a fixed set of answer labels. Results show our approach to be superior for those questions classified as having a miscellaneous answer type.

1 Introduction

Given a question, people are usually able to form an expectation about the type of the answer, even if they do not know the actual answer. An accurate expectation of the answer type makes it much easier to select the answer from a sentence that contains the query words. Consider the question "What is the capital of Norway?" We would expect the answer to be a city and could filter out most of the words in the following sentence:

> The landed aristocracy was virtually crushed by Hakon V, who reigned from 1299 to 1319, and Oslo became the capital of Norway, replacing Bergen as the principal city of the kingdom.

The goal of answer typing is to determine whether a word's semantic type is appropriate as an answer for a question. Many previous approaches to answer typing, e.g., (Ittycheriah et al., 2001; Li and Roth, 2002; Krishnan et al., 2005), employ a predefined set of answer types and use supervised learning or manually constructed rules to classify a question according to expected answer type. A disadvantage of this approach is that there will always be questions whose answers do not belong to any of the predefined types.

Consider the question: "What are tourist attractions in Reims?" The answer may be many things: a church, a historic residence, a park, a famous intersection, a statue, etc. A common method to deal with this problem is to define a catch-all class. This class, however, tends not to be as effective as other answer types.

Another disadvantage of predefined answer types is with regard to granularity. If the types are too specific, they are more difficult to tag. If they are too general, too many candidates may be identified as having the appropriate type.

In contrast to previous approaches that use a supervised classifier to categorize questions into a predefined set of types, we propose an unsupervised method to dynamically construct a probabilistic answer type model for each question. Such a model can be used to evaluate whether or not a word fits into the question context. For example, given the question "What are tourist attractions in Reims?", we would expect the appropriate answers to fit into the context "X is a tourist attraction." From a corpus, we can find the words that appeared in this context, such as:

> A-Ama Temple, Aborigine, addition, Anak Krakatau, archipelago, area, baseball, Bletchley Park, brewery, cabaret, Cairo, Cape Town, capital, center, ...

Using the frequency counts of these words in the context, we construct a probabilistic model to compute $P(in(w,\Gamma)|w)$, the probability for a word w to occur in a set of contexts Γ, given an occurrence of w. The parameters in this model are obtained from a large, automatically parsed, unlabeled corpus. By asking whether a word would occur in a particular context extracted from a question, we avoid explicitly specifying a list of possible answer types. This has the added benefit of being easily adapted to different domains and corpora in which a list of explicit possible answer types may be difficult to enumerate and/or identify within the text.

The remainder of this paper is organized as follows. Section 2 discusses the work related to answer typing. Section 3 discusses some of the key concepts employed by our probabilistic model, including word clusters and the contexts of a question and a word. Section 4 presents our probabilistic model for answer typing. Section 5 compares the performance of our model with that of an oracle and a semi-automatic system performing the same task. Finally, the concluding remarks in are made in Section 6.

2 Related Work

Light et al. (2001) performed an analysis of the effect of multiple answer type occurrences in a sentence. When multiple words of the same type appear in a sentence, answer typing with fixed types must assign each the same score. Light et al. found that even with perfect answer sentence identification, question typing, and semantic tagging, a system could only achieve 59% accuracy over the TREC-9 questions when using their set of 24 non-overlapping answer types. By computing the probability of an answer candidate occurring in the question contexts directly, we avoid having multiple candidates with the same level of appropriateness as answers.

There have been a variety of approaches to determine the answer types, which are also known as *Qtargets* (Echihabi et al., 2003). Most previous approaches classify the answer type of a question as one of a set of predefined types.

Many systems construct the classification rules manually (Cui et al., 2004; Greenwood, 2004; Hermjakob, 2001). The rules are usually triggered by the presence of certain words in the question. For example, if a question contains "author" then the expected answer type is Person.

The number of answer types as well as the number of rules can vary a great deal. For example, (Hermjakob, 2001) used 276 rules for 122 answer types. Greenwood (2004), on the other hand, used 46 answer types with unspecified number of rules.

The classification rules can also be acquired with supervised learning. Ittycheriah, et al. (2001) describe a maximum entropy based question classification scheme to classify each question as having one of the MUC answer types. In a similar experiment, Li & Roth (2002) train a question classifier based on a modified version of SNoW using a richer set of answer types than Ittycheriah et al.

The LCC system (Harabagiu et al., 2003) combines fixed types with a novel loop-back strategy. In the event that a question cannot be classified as one of the fixed entity types or semantic concepts derived from WordNet (Fellbaum, 1998), the answer type model backs off to a logic prover that uses axioms derived form WordNet, along with logic rules, to justify phrases as answers. Thus, the LCC system is able to avoid the use of a miscellaneous type that often exhibits poor performance. However, the logic prover must have sufficient evidence to link the question to the answer, and general knowledge must be encoded as axioms into the system. In contrast, our answer type model derives all of its information automatically from unannotated text.

Answer types are often used as filters. It was noted in (Radev et al., 2002) that a wrong guess about the answer type reduces the chance for the system to answer the question correctly by as much as 17 times. The approach presented here is less brittle. Even if the correct candidate does not have the highest likelihood according to the model, it may still be selected when the answer extraction module takes into account other factors such as the proximity to the matched keywords. Furthermore, a probabilistic model makes it easier to integrate the answer type scores with scores computed by other components in a question answering system in a principled fashion.

3 Resources

Before introducing our model, we first describe the resources used in the model.

3.1 Word Clusters

Natural language data is extremely sparse. Word clusters are a way of coping with data sparseness by abstracting a given word to a class of related words. Clusters, as used by our probabilistic answer typing system, play a role similar to that of named entity types. Many methods exist for clustering, e.g., (Brown et al., 1990; Cutting et al., 1992; Pereira et al., 1993; Karypis et al., 1999).

We used the Clustering By Committee (CBC)

Table 1: Words and their clusters

Word	Clusters
suite	software, network, wireless, ...
	rooms, bathrooms, restrooms, ...
	meeting room, conference room, ...
ghost	rabbit, squirrel, duck, elephant, frog, ...
	goblins, ghosts, vampires, ghouls, ...
	punk, reggae, folk, pop, hip-pop, ...
	huge, larger, vast, significant, ...
	coming-of-age, true-life, ...
	clouds, cloud, fog, haze, mist, ...

algorithm (Pantel and Lin, 2002) on a 10 GB English text corpus to obtain 3607 clusters. The following is an example cluster generated by CBC:

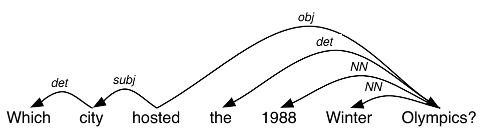
> tension, anger, anxiety, tensions, frustration, resentment, uncertainty, confusion, conflict, discontent, insecurity, controversy, unease, bitterness, dispute, disagreement, nervousness, sadness, despair, animosity, hostility, outrage, discord, pessimism, anguish, ...

In the clustering generated by CBC, a word may belong to multiple clusters. The clusters to which a word belongs often represent the senses of the word. Table 1 shows two example words and their clusters.

3.2 Contexts

The context in which a word appears often imposes constraints on the semantic type of the word. This basic idea has been exploited by many proposals for distributional similarity and clustering, e.g., (Church and Hanks, 1989; Lin, 1998; Pereira et al., 1993).

Similar to Lin and Pantel (2001), we define the contexts of a word to be the undirected paths in dependency trees involving that word at either the beginning or the end. The following diagram shows an example dependency tree:



The links in the tree represent dependency relationships. The direction of a link is from the head to the modifier in the relationship. Labels associated with the links represent types of relations.

In a context, the word itself is replaced with a variable X. We say a word is the filler of a context if it replaces X. For example, the contexts for the word "Olympics" in the above sentence include the following paths:

Context of "Olympics"	Explanation
X $\xrightarrow{NN}$ Winter	Winter X
X $\xrightarrow{NN}$ 1988	1988 X
X $\xleftarrow{obj}$ host	host X
X $\xleftarrow{obj}$ host $\xrightarrow{subj}$ city	city hosted X

In these paths, words are reduced to their root forms and proper names are reduced to their entity tags (we used MUC7 named entity tags).

Paths allow us to balance the specificity of contexts and the sparseness of data. Longer paths typically impose stricter constraints on the slot fillers. However, they tend to have fewer occurrences, making them more prone to errors arising from data sparseness. We have restricted the path length to two (involving at most three words) and require the two ends of the path to be nouns.

We parsed the AQUAINT corpus (3GB) with Minipar (Lin, 2001) and collected the frequency counts of words appearing in various contexts. Parsing and database construction is performed off-line as the database is identical for all questions. We extracted 527,768 contexts that appeared at least 25 times in the corpus. An example context and its fillers are shown in Figure 1.

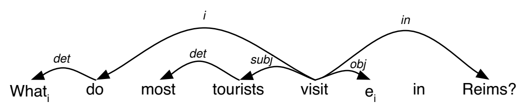
X $\xleftarrow{subj}$ host $\xrightarrow{obj}$ Olympics

Africa 2	grant 1	readiness 2
AP 1	he 2	Rio de Janeiro 1
Argentina 1	homeland 3	Rome 1
Athens 16	IOC 1	Salt Lake City 2
Atlanta 3	Iran 2	school 1
Bangkok 1	Jakarta 1	S. Africa 1
...	...	...
decades 1	president 2	Zakopane 4
facility 1	Pusan 1	
government 1	race 1	

Figure 1: An example context and its fillers

3.2.1 Question Contexts

To build a probabilistic model for answer typing, we extract a set of contexts, called question contexts, from a question. An answer is expected to be a plausible filler of the question contexts. Question contexts are extracted from a question with two rules. First, if the wh-word in a question has a trace in the parse tree, the question contexts are the contexts of the trace. For example, the question “What do most tourists visit in Reims?” is parsed as:



The symbol $\mathrm{e_i}$ is the trace of $\mathrm{what_i}$. Minipar generates the trace to indicate that the word what is the object of visit in the deep structure of the sentence. The following question contexts are extracted from the above question:

Context	Explanation
$X \xleftarrow{obj} \text{visit} \xrightarrow{subj} \text{tourist}$	tourist visits X
$X \xleftarrow{obj} \text{visit} \xrightarrow{in} \text{Reims}$	visit X in Reims

The second rule deals with situations where the wh-word is a determiner, as in the question “Which city hosted the 1988 Winter Olympics?” (the parse tree for which is shown in section 3.2). In such cases, the question contexts consist of a single context involving the noun that is modified by the determiner. The context for the above sentence is $X \xleftarrow{subj} \text{city}$, corresponding to the sentence “X is a city.” This context is used because the question explicitly states that the desired answer is a city. The context overrides the other contexts because the question explicitly states the desired answer type. Experimental results have shown that using this context in conjunction with other contexts extracted from the question produces lower performance than using this context alone.

In the event that a context extracted from a question is not found in the database, we shorten the context in one of two ways. We start by replacing the word at the end of the path with a wildcard that matches any word. If this fails to yield entries in the context database, we shorten the context to length one and replace the end word with automatically determined similar words instead of a wildcard.

3.2.2 Candidate Contexts

Candidate contexts are very similar in form to question contexts, save for one important difference. Candidate contexts are extracted from the parse trees of the answer candidates rather than the question. In natural language, some words may be polysemous. For example, Washington may refer to a person, a city, or a state. The occurrences of Washington in “Washington’s descendants” and “suburban Washington” should not be given the same score when the question is seeking a location. Given that the sense of a word is largely determined by its local context (Choueka and Lusignan, 1985), candidate contexts allow the model to take into account the candidate answers’ senses implicitly.

4 Probabilistic Model

The goal of an answer typing model is to evaluate the appropriateness of a candidate word as an answer to the question. If we assume that a set of answer candidates is provided to our model by some means (e.g., words comprising documents extracted by an information retrieval engine), we wish to compute the value $P(in(w, \Gamma_Q)|w)$. That is, the appropriateness of a candidate answer w is proportional to the probability that it will occur in the question contexts Γ_Q extracted from the question.

To mitigate data sparseness, we can introduce a hidden variable C that represents the clusters to which the candidate answer may belong. As a candidate may belong to multiple clusters, we obtain:

$$\begin{aligned} P(in(w,\Gamma_Q)|w) &= \sum_C P(in(w,\Gamma_Q), C|w) && (1) \\ &= \sum_C P(C|w) P(in(w,\Gamma_Q)|C,w) && (2) \end{aligned}$$

Given that a word appears, we assume that it has the same probability to appear in a context as all other words in the same cluster. Therefore:

$$P(in(w,\Gamma_Q)|C,w) \approx P(in(C,\Gamma_Q)|C) \quad (3)$$

We can now rewrite the equation in (2) as:

$$P(in(w,\Gamma_Q)|w) \approx \sum_C P(C|w) P(in(C,\Gamma_Q)|C) \quad (4)$$

This equation splits our model into two parts: one models which clusters a word belongs to and the other models how appropriate a cluster is to the question contexts. When Γ_Q consists of multiple contexts, we make the naïve Bayes assumption that each individual context $\gamma_Q \in \Gamma_Q$ is independent of all other contexts given the cluster C.

$$P(in(w,\Gamma_Q)|w) \approx \sum_C P(C|w) \prod_{\gamma_Q \in \Gamma_Q} P(in(C,\gamma_Q)|C) \quad (5)$$

Equation (5) needs the parameters $P(C|w)$ and $P(in(C,\gamma_Q)|C)$, neither of which are directly available from the context-filler database. We will discuss the estimation of these parameters in Section 4.2.

4.1 Using Candidate Contexts

The previous model assigns the same likelihood to every instance of a given word. As we noted in section 3.2.2, a word may be polysemous. To take into account a word's context, we can instead compute $P(in(w, \Gamma_Q)|w, in(w, \Gamma_w))$, where Γ_w is the set of contexts for the candidate word w in a retrieved passage.

By introducing word clusters as intermediate variables as before and making a similar assumption as in equation (3), we obtain:

$$\begin{aligned} & P(in(w, \Gamma_Q)|w, in(w, \Gamma_w)) \\ &= \sum_C P(in(w, \Gamma_Q), C|w, in(w, \Gamma_w)) \quad (6) \\ &\approx \sum_C P(C|w, in(w, \Gamma_w)) P(in(C, \Gamma_Q)|C) \quad (7) \end{aligned}$$

Like equation (4), equation (7) partitions the model into two parts. Unlike $P(C|w)$ in equation (4), the probability of the cluster is now based on the particular occurrence of the word in the candidate contexts. It can be estimated by:

$$\begin{aligned} & P(C|w, in(w, \Gamma_w)) \\ &= \frac{P(in(w, \Gamma_w)|w, C) P(w, C)}{P(in(w, \Gamma_w)|w) P(w)} \quad (8) \\ &\approx \frac{\prod_{\gamma_w \in \Gamma_w} P(in(w, \gamma_w)|w, C)}{\prod_{\gamma_w \in \Gamma_w} P(in(w, \gamma_w)|w)} \times P(C|w) \quad (9) \\ &= \prod_{\gamma_w \in \Gamma_w} \left(\frac{P(C|w, in(w, \gamma_w))}{P(C|w)} \right) \times P(C|w) \quad (10) \end{aligned}$$

4.2 Estimating Parameters

Our probabilistic model requires the parameters $P(C|w)$, $P(C|w, in(w, \gamma))$, and $P(in(C, \gamma)|C)$, where w is a word, C is a cluster that w belongs to, and γ is a question or candidate context. This section explains how these parameters are estimated without using labeled data.

The context-filler database described in Section 3.2 provides the joint and marginal frequency counts of contexts and words ($|in(\gamma, w)|$, $|in(*, \gamma)|$ and $|in(w, *)|$). These counts allow us to compute the probabilities $P(in(w, \gamma))$, $P(in(w, *))$, and $P(in(*, \gamma))$. We can also compute $P(in(w, \gamma)|w)$, which is smoothed with add-one smoothing (see equation (11) in Figure 2). The estimation of $P(C|w)$ presents a challenge. We have no corpus from which we can directly measure $P(C|w)$ because word instances are not labeled with their clusters.

$$P(in(w, \gamma)|w) = \frac{|in(w, \gamma)| + P(in(*, \gamma))}{|in(w, *)| + 1} \quad (11)$$

$$P_u(C|w) = \begin{cases} \frac{1}{|\{C'|w \in C'\}|} & \text{if } w \in C, \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

$$P(C|w) = \frac{\sum_{w' \in S(w)} sim(w, w') \times P_u(C|w')}{\sum_{\substack{\{C'|w \in C'\}, \\ w' \in S(w)}} sim(w, w') \times P_u(C'|w')} \quad (13)$$

$$P(in(C, \gamma)|C) = \frac{\sum_{w' \in C} P(C|w') \times |in(w', \gamma)| + P(in(*, \gamma))}{\sum_{w' \in C} P(C|w') \times |in(w', *)| + 1} \quad (14)$$

Figure 2: Probability estimation

We use the average weighted "guesses" of the top similar words of w to compute $P(C|w)$ (see equation 13). The intuition is that if w' and w are similar words, $P(C|w')$ and $P(C|w)$ tend to have similar values. Since we do not know $P(C|w')$ either, we substitute it with uniform distribution $P_u(C|w')$ as in equation (12) of Figure 2. Although $P_u(C|w')$ is a very crude guess, the weighted average of a set of such guesses can often be quite accurate.

The similarities between words are obtained as a byproduct of the CBC algorithm. For each word, we use $S(w)$ to denote the top-n most similar words (n=50 in our experiments) and $sim(w, w')$ to denote the similarity between words w and w'. The following is a sample similar word list for the word suit:

> S(suit) = {lawsuit 0.49, suits 0.47, complaint 0.29, lawsuits 0.27, jacket 0.25, countersuit 0.24, counterclaim 0.24, pants 0.24, trousers 0.22, shirt 0.21, slacks 0.21, case 0.21, pantsuit 0.21, shirts 0.20, sweater 0.20, coat 0.20, ...}

The estimation for $P(C|w, in(w, \gamma_w))$ is similar to that of $P(C|w)$ except that instead of all $w' \in S(w)$, we instead use $\{w'|w' \in S(w) \wedge in(w', \gamma_w)\}$. By only looking at a particular context γ_w, we may obtain a different distribution over C than $P(C|w)$ specifies. In the event that the data are too sparse to estimate $P(C|w, in(w, \gamma_w))$, we fall back to using $P(C|w)$.

$P(in(C, \gamma)|C)$ is computed in (14) by assuming each instance of w contains a fractional instance of C and the fractional count is $P(C|w)$. Again, add-one smoothing is used.

System	Median %	Top 1%	Top 5%	Top 10%	Top 50%
Oracle	0.7%	89 (57%)	123 (79%)	131 (85%)	154 (99%)
Frequency	7.7%	31 (20%)	67 (44%)	86 (56%)	112 (73%)
Our model	**1.2%**	**71 (46%)**	**106 (69%)**	**119 (77%)**	**146 (95%)**
no cand. contexts	2.2%	58 (38%)	102 (66%)	113 (73%)	145 (94%)
ANNIE	4.0%	54 (35%)	79 (51%)	93 (60%)	123 (80%)

Table 2: Summary of Results

5 Experimental Setup & Results

We evaluate our answer typing system by using it to filter the contents of documents retrieved by the information retrieval portion of a question answering system. Each answer candidate in the set of documents is scored by the answer typing system and the list is sorted in descending order of score. We treat the system as a filter and observe the proportion of candidates that must be accepted by the filter so that at least one correct answer is accepted. A model that allows a low percentage of candidates to pass while still allowing at least one correct answer through is favorable to a model in which a high number of candidates must pass. This represents an intrinsic rather than extrinsic evaluation (Mollá and Hutchinson, 2003) that we believe illustrates the usefulness of our model.

The evaluation data consist of 154 questions from the TREC-2003 QA Track (Voorhees, 2003) satisfying the following criteria, along with the top 10 documents returned for each question as identified by NIST using the PRISE[1] search engine.

- the question begins with What, Which, or Who. We restricted the evaluation such questions because our system is designed to deal with questions whose answer types are often semantically open-ended noun phrases.
- There exists entry for the question in the answer patterns provided by Ken Litkowski[2].
- One of the top-10 documents returned by PRISE contains a correct answer.

We compare the performance of our probabilistic model with that of two other systems. Both comparison systems make use of a small, predefined set of manually-assigned MUC-7 named-entity types (location, person, organization, cardinal, percent, date, time, duration, measure, money) augmented with thing-name (proper names of inanimate objects) and miscellaneous (a catch-all answer type of all other candidates). Some examples of thing-name are *Guinness Book of World Records*, *Thriller*, *Mars Pathfinder*, and *Grey Cup*. Examples of miscellaneous answers are *copper*, *oil*, *red*, and *iris*.

The differences in the comparison systems is with respect to how entity types are assigned to the words in the candidate documents. We make use of the ANNIE (Maynard et al., 2002) named entity recognition system, along with a manual assigned "oracle" strategy, to assign types to candidate answers. In each case, the score for a candidate is either 1 if it is tagged as the same type as the question or 0 otherwise. With this scoring scheme producing a sorted list we can compute the probability of the first correct answer appearing at rank $R = k$ as follows:

$$P(R = k) = \prod_{i=0}^{k-2} \left(\frac{t - c - i}{t - i} \right) \frac{c}{t - k + 1} \qquad (15)$$

where t is the number of unique candidate answers that are of the appropriate type and c is the number of unique candidate answers that are correct.

Using the probabilities in equation (15), we compute the expected rank, $E(R)$, of the first correct answer of a given question in the system as:

$$E(R) = \sum_{k=1}^{t-c+1} kP(R = k) \qquad (16)$$

Answer candidates are the set of ANNIE-identified tokens with stop words and punctuation removed. This yields between 900 and 8000 candidates for each question, depending on the top 10 documents returned by PRISE. The oracle system represents an upper bound on using the predefined set of answer types. The ANNIE system represents a more realistic expectation of performance.

The median percentage of candidates that are accepted by a filter over the questions of our evaluation data provides one measure of performance and is preferred to the average because of the effect of large values on the average. In QA, a system accepting 60% of the candidates is not significantly better or worse than one accepting 100%,

[1]www.itl.nist.gov/iad/894.02/works/papers/zp2/zp2.html

[2]trec.nist.gov/data/qa/2003_qadata/03QA.tasks/t12.pats.txt

System	Measure	Question Type						
		All (154)	Location (57)	Person (17)	Organization (19)	Thing-Name (17)	Misc (37)	Other (7)
Our model	Median	1.2%	0.8%	2.0%	1.3%	3.7%	3.5%	12.2%
	Top 1%	71	34	6	9	7	13	2
	Top 5%	106	53	11	11	10	19	2
	Top 10%	119	55	12	17	10	22	3
	Top 50%	146	56	16	18	17	34	5
Oracle	Median	0.7%	0.4%	1.0%	0.3%	0.4%	16.0%	0.3%
	Top 1%	89	44	8	16	14	1	6
	Top 5%	123	57	17	19	17	6	7
	Top 10%	131	57	17	19	17	14	7
	Top 50%	154	57	17	19	17	37	7
ANNIE	Median	4.0%	0.6%	1.4%	6.1%	100%	16.7%	50.0%
	Top 1%	54	39	5	7	0	0	3
	Top 5%	79	53	12	9	0	2	3
	Top 10%	93	54	13	11	0	12	3
	Top 50%	123	56	16	15	5	28	3

Table 3: Detailed breakdown of performance

but the effect on average is quite high. Another measure is to observe the number of questions with at least one correct answer in the top $N\%$ for various values of N. By examining the number of correct answers found in the top $N\%$ we can better understand what an effective cutoff would be.

The overall results of our comparison can be found in Table 2. We have added the results of a system that scores candidates based on their frequency within the document as a comparison with a simple, yet effective, strategy. The second column is the median percentage of where the highest scored correct answer appears in the sorted candidate list. Low percentage values mean the answer is usually found high in the sorted list. The remaining columns list the number of questions that have a correct answer somewhere in the top $N\%$ of their sorted lists. This is meant to show the effects of imposing a strict cutoff prior to running the answer type model.

The oracle system performs best, as it benefits from both manual question classification and manual entity tagging. If entity assignment is performed by an automatic system (as it is for ANNIE), the performance drops noticeably. Our probabilistic model performs better than ANNIE and achieves approximately 2/3 of the performance of the oracle system. Table 2 also shows that the use of candidate contexts increases the performance of our answer type model.

Table 3 shows the performance of the oracle system, our model, and the ANNIE system broken down by manually-assigned answer types. Due to insufficient numbers of questions, the cardinal, percent, time, duration, measure, and money types are combined into an "Other" category. When compared with the oracle system, our model performs worse overall for questions of all types except for those seeking miscellaneous answers. For miscellaneous questions, the oracle identifies all tokens that do not belong to one of the other known categories as possible answers. For all questions of non-miscellaneous type, only a small subset of the candidates are marked appropriate. In particular, our model performs worse than the oracle for questions seeking persons and thing-names. Person questions often seek rare person names, which occur in few contexts and are difficult to reliably cluster. Thing-name questions are easy for a human to identify but difficult for automatic system to identify. Thing-names are a diverse category and are not strongly associated with any identifying contexts.

Our model outperforms the ANNIE system in general, and for questions seeking organizations, thing-names, and miscellaneous targets in particular. ANNIE may have low coverage on organization names, resulting in reduced performance. Like the oracle, ANNIE treats all candidates not assigned one of the categories as appropriate for miscellaneous questions. Because ANNIE cannot identify thing-names, they are treated as miscellaneous. ANNIE shows low performance on thing-names because words incorrectly assigned types are sorted to the bottom of the list for miscellaneous and thing-name questions. If a correct answer is incorrectly assigned a type it will be sorted near the bottom, resulting in a poor score.

6 Conclusions

We have presented an unsupervised probabilistic answer type model. Our model uses contexts derived from the question and the candidate answer to calculate the appropriateness of a candidate answer. Statistics gathered from a large corpus of text are used in the calculation, and the model is constructed to exploit these statistics without being overly specific or overly general.

The method presented here avoids the use of an explicit list of answer types. Explicit answer types can exhibit poor performance, especially for those questions not fitting one of the types. They must also be redefined when either the domain or corpus substantially changes. By avoiding their use, our answer typing method may be easier to adapt to different corpora and question answering domains (such as bioinformatics).

In addition to operating as a stand-alone answer typing component, our system can be combined with other existing answer typing strategies, especially in situations in which a catch-all answer type is used. Our experimental results show that our probabilistic model outperforms the oracle and a system using automatic named entity recognition under such circumstances. The performance of our model is better than that of the semi-automatic system, which is a better indication of the expected performance of a comparable real-world answer typing system.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments on improving the paper. The first author is supported by the Natural Sciences and Engineering Research Council of Canada, the Alberta Ingenuity Fund, and the Alberta Informatics Circle of Research Excellence.

References

P.F. Brown, V.J. Della Pietra, P.V. deSouza, J.C. Lai, and R.L. Mercer. 1990. Class-based n-gram Models of Natural Language. *Computational Linguistics*, 16(2):79–85.

Y. Choueka and S. Lusignan. 1985. Disambiguation by Short Contexts. *Computer and the Humanities*, 19:147–157.

K. Church and P. Hanks. 1989. Word Association Norms, Mutual Information, and Lexicography. In *Proceedings of ACL-89*, pages 76–83, Vancouver, British Columbia, Canada.

H. Cui, K. Li, R. Sun, T-S. Chua, and M-K. Kan. 2004. National University of Singapore at the TREC-13 Question Answering Main Task. In *Notebook of TREC 2004*, pages 34–42, Gaithersburg, Maryland.

D.R. Cutting, D. Karger, J. Pedersen, and J.W. Tukey. 1992. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. In *Proceedings of SIGIR-92*, pages 318–329, Copenhagen, Denmark.

A. Echihabi, U. Hermjakob, E. Hovy, D. Marcu, E. Melz, and D. Ravichandran. 2003. Multiple-Engine Question Answering in TextMap. In *Proceedings of TREC 2003*, pages 772–781, Gaithersburg, Maryland.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Massachusetts.

M.A. Greenwood. 2004. AnswerFinder: Question Answering from your Desktop. In *Proceedings of the Seventh Annual Colloquium for the UK Special Interest Group for Computational Linguistics (CLUK '04)*, University of Birmingham, UK.

S. Harabagiu, D. Moldovan, C. Clark, M. Bowden, J. Williams, and J. Bensley. 2003. Answer Mining by Combining Extraction Techniques with Abductive Reasoning. In *Proceedings of TREC 2003*, pages 375–382, Gaithersburg, Maryland.

U. Hermjakob. 2001. Parsing and Question Classification for Question Answering. In *Proceedings of the ACL Workshop on Open-Domain Question Answering*, Toulouse, France.

A. Ittycheriah, M. Franz, W-J. Zhu, and A. Ratnaparkhi. 2001. Question Answering Using Maximum Entropy Components. In *Proceedings of NAACL 2001*, Pittsburgh, Pennsylvania.

G. Karypis, E.-H. Han, and V. Kumar. 1999. Chameleon: A Hierarchical Clustering Algorithm using Dynamic Modeling. *IEEE Computer: Special Issue on Data Analysis and Mining*, 32(8):68–75.

V. Krishnan, S. Das, and S. Chakrabarti. 2005. Enhanced Answer Type Inference from Questions using Sequential Models. In *Proceedings of HLT/EMNLP 2005*, pages 315–322, Vancouver, British Columbia, Canada.

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

M. Light, G. Mann, E. Riloff, and E. Breck. 2001. Analyses for Elucidating Current Question Answering Technology. *Natural Language Engineering*, 7(4):325–342.

D. Lin and P. Pantel. 2001. Discovery of Inference Rules for Question Answering. *Natural Language Engineering*, 7(4):343–360.

D. Lin. 1998. Automatic Retrieval and Clustering of Similar Words. In *Proceedings of COLING-ACL 1998*, Montreal, Québec, Canada.

D. Lin. 2001. Language and Text Analysis Tools. In *Proceedings of HLT 2001*, pages 222–227, San Diego, California.

D. Maynard, V. Tablan, H. Cunningham, C. Ursu, H. Saggion, K. Bontcheva, and Y. Wilks. 2002. Architectural Elements of Language Engineering Robustness. *Natural Language Engineering*, 8(2/3):257–274.

D. Mollá and B. Hutchinson. 2003. Intrinsic versus Extrinsic Evaluations of Parsing Systems. In *Proceedings of EACL Workshop on Evaluation Initiatives in Natural Language Processing*, pages 43–50, Budapest, Hungary.

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

F. Pereira, N. Tishby, and L. Lee. 1993. Distributional Clustering of English Words. In *Proceedings of ACL 1992*, pages 183–190.

D. Radev, W. Fan, H. Qi, H. Wu, and A. Grewal. 2002. Probablistic Question Answering on the Web. In *Proceedings of the Eleventh International World Wide Web Conference*.

E.M. Voorhees. 2003. Overview of the TREC 2003 Question Answering Track. In *Proceedings of TREC 2003*, Gaithersburg, Maryland.