Automatic Answer Typing for How-Questions

Christopher Pinchak and **Shane Bergsma**
Department of Computing Science
University of Alberta
Edmonton, Alberta, T6G 2E8, Canada
{pinchak,bergsma}@cs.ualberta.ca

Abstract

We introduce an answer typing strategy specific to quantifiable *how* questions. Using the web as a data source, we automatically collect answer units appropriate to a given how-question type. Experimental results show answer typing with these units outperforms traditional fixed-category answer typing and other strategies based on the occurrences of numerical entities in text.

1 Introduction

Question answering (QA) systems are emerging as a viable means of obtaining specific information in the face of large availability. Answer typing is an important part of QA because it allows the system to greatly reduce the number of potential answers, using general knowledge of the answer form for a specific question. For example, for *what*, *where*, and *who* questions like "What is the capital of Canada?", answer typing can filter the phrases which might be proposed as candidate answers, perhaps only identifying those textual entities known to be cities.

We focus on answer typing for *how*-questions, a subset of questions which have received little specific attention in the QA community. Rather than seeking an open-ended noun or verb phrase, how-questions often seek a numerical measurement expressed in terms of a certain kind of unit, as in the following example:

Example 1: "How heavy is a grizzly bear?"

An answer typing system might expect answers to include units like *kilograms*, *pounds*, or *tons*. Entities with inappropriate units, such as *feet*, *meters*, or *honey pots*, would be excluded as candidate answers.

We specifically handle the subset of how-questions that we call *how-adjective* questions; that is, questions of the form "How *adjective*...?" such as Example 1. In particular, we do not address "how many" questions, which usually specify the units directly following *many*, nor "how much" questions, which generally seek a monetary value.

Hand-crafting a comprehensive list of units appropriate to many different adjectives is time-consuming and likely to miss important units. For example, an annotator might miss *gigabytes* for a measure of "how large." Instead of compiling a list manually, we propose a means of automatically generating lists of appropriate units for a number of real-world questions.

How-adjective questions represent a significant portion of queries sent to search engines; of the 35 million queries in the AOL search query data set (Pass et al., 2006), over 11,000 are of the form "how *adjective*..." – close to one in every three thousand queries. Of those 11,000 queries, 152 different adjectives are used, ranging from the expected "how old" and "how far" to the obscure "how orwellian."

This high proportion of queries is especially striking given that search engines provide little support for answering how-adjective questions. Indeed, most IR systems work by keyword matching. Entering Example 1 into a search engine returns documents discussing the grizzly's "heavy fur," "heavy, shaggy coat" and "heavy stout body." When faced

Proceedings of NAACL HLT 2007, pages 516–523,
Rochester, NY, April 2007.

with such results, a smart search engine user knows to inject answer units into their query to refine their search, perhaps querying "grizzly pounds." They may also convert their adjective (*heavy*) to a related concept (*weight*), for the query "grizzly weight."

Similarly, our approach discovers unit types by first converting the adjective to a related concept, using information in a structured ontology. For example, "big" can be used to obtain "size," and "tall" can derive "height." We then use an online search engine to automatically find units appropriate to the concept, given the assumption that the concept is explicitly measured in terms of specific units, e.g., height can be measured in feet, weight can be measured in pounds, and size can be measured in gigabytes.

By automatically extracting units, we do not require a set of prior questions with associated answers. Instead, we use actual questions as a source of realistic adjectives only. This is important because while large sets of existing questions can be obtained (Li and Roth, 2002), there are many fewer questions with available answers.

Our experiments demonstrate that how-question-specific unit lists consistently achieve higher answer identification performance than fixed-type, general-purpose answer typing (which propose all numerical entities as answer candidates). Furthermore, our precomputed, automatically-generated unit lists are shown to consistently achieve better performance than baseline systems which derive unit lists at runtime from documents relevant to the answer query, even when such documents are gathered using perfect knowledge of the answer distribution.

The outline of the paper is as follows. In Section 2 we outline related work. In Section 3 we provide the framework of our answer-typing model. Section 4 describes the implementation details of the model. Section 5 describes our experimental methodology, while Section 6 shows the benefits of using automatic how-question answer-typing. We conclude with possible directions of future research opened by this novel problem formulation.

2 Previous Work

Answer typing is an important component of any QA system, but varies greatly in the approach taken (Prager et al., 2003; Harabagiu et al., 2005). Basically, answer typing provides a means of filtering answer candidates as either appropriate or inappropriate to the question. For example, Li and Roth (2002) assign one of fifty possible types to a question based on features present in the question. Answer candidates can then be selected from text by finding entities whose type matches that of the input question. Similarly, Ittycheriah et al. (2000) assign one of the MUC named-entity types to each input question. In these fixed-category approaches, how-questions are assigned a fixed type in the same manner as other questions. For how-questions, this corresponds to a numerical type. However, retrieving all numerical entities will provide lower answer identification precision than a system that only provides those specified with the expected answer units.

Pinchak and Lin (2006) propose a dynamic answer typing system which computes a unique score for the appropriateness of any word to a particular question. Unfortunately, their question context-mapping is limited to *what*, *where*, and *who* questions, and thus is not defined for how-questions.

Wu et al. (2005) handle how-questions differently than other questions. They use special hand-crafted rules to assign a particular answer target during the answer typing phase. In this way, they take advantage of the structure inherent in how-questions rather than just treating them as general queries. However, manually hand-crafting types is costly, and would have to be repeated if the system was moved to a new language or a new query domain. Our automatic approach does not suffer from this drawback.

Light et al. (2001) showed that for a small fixed set of answer types, multiple words tagged with the same type will exist even with perfect passage retrieval, sentence retrieval, and type assignment. For example, Example 1 may be answered with a sentence such as "bears range in weight from the smaller black bear at 400 pounds to the gigantic grizzly at over 1200 pounds" in which two answers have appropriate units but only one of which is correct. We provide results in Section 6 confirming the limits of answer typing at narrowing answer focus, using varying levels of perfect information.

Our approach makes use of the web as a large corpus of useful information. Exploiting the vast amount of data on the web is part of a growing trend in Natural Language Processing (Keller and Lapata,

2003). Indeed, many QA systems have been developed using the web (to varying degrees) to assist in finding a correct answer (Brill et al., 2001; Cucerzan and Agichtein, 2005; Radev et al., 2001), as the web is the largest available corpus even if its information can be difficult to harness. Rather than relying on the web to find the answer to a question, we rely on it as a source of information on appropriate units only. Should the domain of the question answering system change from general factoid questions, units may be extracted from a smaller, domain-specific corpus.

3 Model Framework

The objective of our model is to create a list of relevant units for an adjective that may be found in a how-question. We wish to create these lists *a priori* and off-line so that they are applicable to future questions. Although the model described here can be applied on-line at the time of question answering, the resources and time required make off-line generation of unit lists the preferred approach.

We wish to automatically learn a mapping $T : A \rightarrow U_A$ in which A is a set of adjectives derived from how-questions and U_A is a set of lists of units associated with these adjectives. For example, an element of this mapping might be:

$$\textit{high} \in A \rightarrow \{\text{feet, meter, foot, inches, ...}\} \in U_A$$

which assigns height measurements to "how high" questions. Inducing this mapping means establishing a connection, or *co-occurrence*, between each adjective a and its units U_a. In the following subsections, we show how to establish this connection.

3.1 Using WordNet for Adjective Expansion

In common documents, such as news articles or web pages, the co-occurrence of an adjective and its units may be unlikely. For example, the co-occurrence between "heavy" and "pounds" may not be as prevalent as the co-occurrence between "weight" and "pounds." We therefore propose using WordNet (Fellbaum, 1998) to expand the how-adjective a to a set of related concepts the adjective may be used to describe. We denote a related concept of a as r. In the above example, "heavy" can be used to describe a "weight." Two useful WordNet relations are the *attribute* relation, in which the adjective is an attribute of the concept, and in cases where no attribute exists, the *derivationally-related* words. "Heavy" is an attribute of "weight" whereas the derivationally-related form is "heaviness," a plausible but less useful concept. Next we describe how the particular co-occurrence of the related concept r and unit u is obtained.

3.2 Using Google to Obtain Counts

We selected the Google search engine as a source of co-occurrence data due to the large number of indexed documents from which co-occurrence counts can be derived. To further enhance the quality of co-occurrence data, we search on the specific phrase "r is measured in" in which r is one of the related concepts of a. This allows for the simultaneous discovery of unknown units and the retrieval of their co-occurrence counts.

Sentences in which the pattern occurs are parsed using Minipar (Lin, 1998b) so that we can obtain the word related to "measured" via the prepositional *in* relation. This allows us to handle sentential constructions that may intervene between "measured" and a meaningful unit. For each unit u that is related to "measured" via *in*, we increment the co-occurrence count $f(u, r)$, thereby collecting frequency counts for each u with r.

The pattern's precision prevents incidental co-occurrence between a related concept and some unit that may occur simply because of the general topic of the document. For example, "size is measured in" matches "Size is measured in gigabytes, and performance is measured in milliseconds". In this example, the co-occurrence count $f(\text{gigabytes}, \text{size})$ would be incremented by one, whereas there is no co-occurrence between "size" and "milliseconds." Due to the large amount of data available to Google, we can afford to restrict ourselves to a single pattern and still expect to find meaningful units.

To gather the co-occurrence counts between an adjective a and a unit u, we first expand a to the set of related concepts R_a and then compute:

$$f(u, a) = \sum_{r \in R_a} f(u, r) \qquad (1)$$

These frequencies can then be used by the scoring functions described in the following section.

3.3 Filtering the Unit List

For a given adjective a and a particular unit u with co-occurrence $f(u, a)$, we define two important statistics:

$$P(u|a) = \frac{f(u,a)}{\sum_{u' \in U} f(u', a)} \quad (2)$$

$$P(a|u) = \frac{f(u,a)}{\sum_{a' \in A} f(u, a')} \quad (3)$$

The first equation measures the likelihood of a unit u being an answer unit for a how-question with the given adjective a. The second equation measures, for a given unit u, how likely a how question with adjective a asked the question answered by u. The second measure is particularly useful in cases where a unit u co-occurs with a number of different adjectives. These units are inherently less useful for answer typing. For example, if the word "terms" occurs on the unit list of adjectives such as "high," "long," and "heavy," it may indicate that "terms" is not an appropriate measure for any of these concepts, but rather just a word likely to co-occur with nouns that can be measured.

We propose using the measures $P(u|a)$ and $P(u|a)P(a|u)$ to score and rank our how-adjective unit lists. $P(a|u)$ alone showed inferior performance on the development set and so will not be further considered. $P(u|a)P(a|u)$ approximates the standard tf-idf measure (Salton and Buckley, 1988). $P(u|a)$ is the term frequency tf in the unit list and $P(a|u)$ is the inverse document frequency idf of the unit over all unit lists. Using these measures, we can create a unit list for an adjective a as

$$U_a = \{u : Score(u, a) \geq \mathcal{T}\} \quad (4)$$

in which $Score(u, a)$ is the score of unit u with adjective a (either $P(u|a)$ or $P(u|a)P(a|u)$) and $\mathcal{T}$ is some threshold imposed to deal with the amount of noise present in the co-occurrence data. This threshold allows us to vary the required strength of the association between the unit and the question in order to consider the unit as appropriate to the how-adjective. In Section 6, we demonstrate this flexibility by showing how answer identification precision and recall can be traded off as desired by the given application. The value $Score(u, a)$ can also be passed to downstream modules of the question answering process (such as the answer extractor), which may then exploit the association value directly.

4 Implementation Details

4.1 Automatic How-Adjective Discovery

An initial step in implementing answer typing for how-adjective questions is to decide which adjectives would benefit from types. WordNet gives a set of all adjectives, but providing answer type units for all these adjectives is unnecessary and potentially misleading. Many adjectives would clearly never occur in a how-adjective query (i.e., "how vehicular...?"), and even some that do, like the "how orwellian" query mentioned above, are difficult to quantify. For these, a simple search with keyword matching as in typical information retrieval would be preferable.

We have a two-stage process for identifying unit-typable how-adjectives. First, we examine the AOL query data (Pass et al., 2006) and extract as candidates all 152 adjectives that occur with the pattern "how *adjective* is/are/was/were." Second, we filter adjectives that do not have a related concept in WordNet (Section 3.1). We built unit lists for the 104 adjectives that remained.

Given that both the query database and WordNet may lack information, it is important to consider the coverage of actual how-adjective questions that unit lists collected this way may have. Reassuringly, experiments have shown 100% coverage of the 96 adjectives in our development and test question set, taken from the TREC QA corpus (see Section 5).

4.2 Similar Word Expansion

Unfortunately, we found that search results obtained using the pattern described in Section 3.2 do not produce a wide variety of units. Web pages often do not use a slang term when mentioning the unit of measurement; a search for "size is measured in gigs" on Google returns zero pages. Also, searching with Google's API and obtaining relevant documents can be time consuming, and we must limit the number of pages considered. Thus, there is strong motivation to expand the list of units obtained from Google by automatically considering *similar* units.

We gather similar units from an automatically-constructed thesaurus of distributionally similar words (Lin, 1998a). The similar word expansion can add a term like *gigs* as a unit for *size* by virtue of its association with *gigabytes*, which is on the original list.

Unit similarity can be thought of as a mapping $S : U \rightarrow V_U$ in which U is a set of units and V_U is sets of related units. If u is an element of U_a for a particular adjective a, the mapping $u \rightarrow V_u$ gives us a way to add new words to the unit list for a. For example, the similar word list for "gigabytes" might be {*GB, megabytes, kilobytes, KB, byte, GHz, gigs...* }, which can all be added to the unit list for the adjective "large."

After expanding each element of the unit list U_a for adjective a, we have a new set of units W_a:

$$W_a = U_a \cup S(U_a) \tag{5}$$

where $S(U_a) = \bigcup_{u \in U_a} V_u$.

For each $v \in V_u$ there is an associated score $\alpha(u, v) \in [0, 1]$ that measures how similar v is to u. We define the score of units that do not co-occur on similar word lists to be zero and the similarity of two identical units to be one. We can then use these scores to assign estimated co-occurrence counts for any unit w in the expanded unit list W_a:

$$\hat{f}(w, a) = \sum_{u \in U_a} \alpha(w, u) f(u, a) \tag{6}$$

If a unit $u \in U_a$ also occurs in the set of expanded similar units for another another unit $u' \in U_a$, that is, $u \in V_{u'}$, then the original co-occurrence frequency of u and a, $\alpha(u, u) f(u, a)$, will be boosted by the similarity-weighted frequency of u on the expanded unit list of u', $\alpha(u, u') f(u', a)$.

4.3 Selection of Answer Candidates

For a given how-adjective question and a document of interest, we use a two-stage process to identify the entities in the document that are suitable answers for the question. First, the named entity recognizer of Minipar is used to identify all numerical entities in text, labeled as *NUM*. Minipar labels times, dates, monetary amounts, and address numbers with types other than *NUM* and so we can correctly exclude these from consideration. We then inspect the context of all *NUM* entities to see if a unit exists on the pre-computed unit list for the given how-adjective. Textual entities that pass both stages of our identification process are considered as candidate answers.

5 Experiments

This section presents experiments comparing our how-adjective answer typing approach to alternative schemes on an answer identification task. We compare our two unit ranking functions $P(u|a)$ and $P(u|a)P(a|u)$ (Section 3.3) and test the merits of using the similar unit expansion (Section 4.2).

5.1 Evaluation Questions

The clearest way to test a QA system is to evaluate it on a large set of questions. Although our answer typing system is not capable of fully answering questions, we will make use of the how-adjective questions from TREC 2002-2005 (Vorhees, 2002) as a set of test data. We take eight of the questions as a development set (used for preliminary investigations of scoring functions – no parameters can be set on the development set specifically) and 86 of the questions as a final, unseen test set. Seventeen different adjectives occur in the test questions.

5.2 Evaluation Methodology

We evaluate our system with an approach we call Answer-Identification Precision Recall (AIPR). For a particular scoring threshold (Section 3.3), each adjective has a corresponding unit list, which is used to extract answer candidates from documents (Section 4.3). To ensure the performance of the IR-engine is not an issue in evaluation, we only use documents judged to contain the correct answer by TREC.

Answer-identification precision corresponds to the number of correct answers among the candidate answers extracted by our system. Answer-identification recall is the number of correct answers extracted among the total number of correct answers in the answer documents.

A plot of AIPR allows the designer of a particular QA system to decide on the optimum PR-tradeoff for the answer typing task. If other stages of QA rely on a large number of candidates, a high recall value may be desired so no potential answers are missed. If answer typing is used as a means of boosting already-likely answers, high precision may instead be favoured.

5.3 Comparison Systems

This section describes the various systems we compare with our approach. Recall that perfect AIPR performance is not possible with typing alone (Section 2, (Light et al., 2001)), and thus we provide some of our comparison systems with varying amounts of perfect answer information in order to establish the highest performance possible in different scenarios on the given dataset.

Query-specific Oracle: The best possible system creates a unit list for each specific how-question individually. This list is created using only those units in the answer pattern of the TREC-provided judgement for this specific question.

Adjective-specific Oracle: This system is designed, like ours, to provide a unit list for each how-adjective, rather than for a specific question. The unit list for a particular adjective contains all the units from all the test set answers of how-adjective questions containing that adjective. It is optimal in the sense it will identify every correct answer for each how-adjective, but only contains those units necessary for this identification.

Fixed-Category: This system gives the performance of a general-purpose, fixed-category answer typing approach applied to how-questions. In a fixed-category approach, all how-questions are classified as seeking numerical answers, and thus all numerical answers are returned as answer candidates.

IR-Document Inferred: Here we infer question units from documents believed to be relevant to the question. An IR system (TREC's PRISE) is given a how-adjective question, and returns a set of documents for that query. Every numerical digit in the documents can be considered a possible answer to the question, and the units associated with those values can be collected as the unit list, ranked (and thresholded) by frequency. We remove units that occur in a list of 527 stopwords, and filter numerical modifiers like "hundred, thousand, million, etc."

Answer-Document Inferred: This approach is identical to the IR-Document Inferred approach, except the documents are only those documents judged by TREC to contain the answer. In this way the Answer-Document Inferred approach provides somewhat of an upper bound on Document Inferred unit typing, by assuming perfect document retrieval.

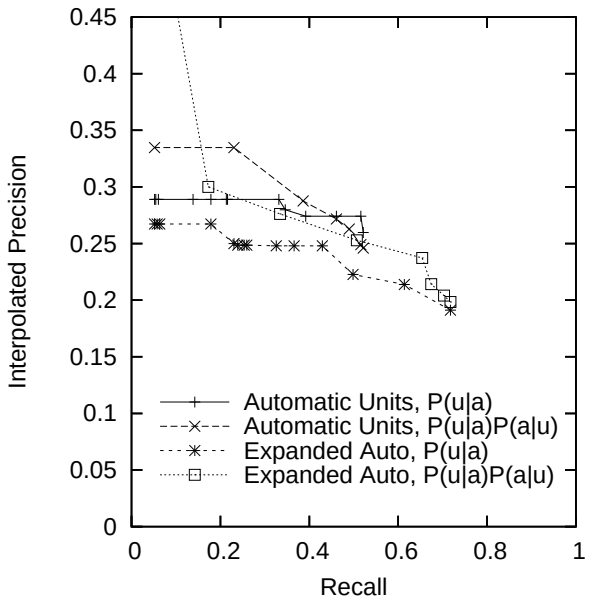

Figure 1: Microaveraged AIPR with different scoring functions, unit lists.

Inferring the answer units from the set of relevant documents is similar in spirit to (Daumé III and Marcu, 2006). In one of their experiments in query-focused summarization, they show competitive summarization performance *without even providing the query*, as the query model is inferred solely from the commonality in relevant documents. In our case, good performance will also be possible if the actual answers have the highest commonality among the numerical values in the relevant documents.

6 Results

The microaveraged Answer-Identification Precision Recall over all question-answer pairs is plotted in Figures 1 and 2. Macroaveraged results are similar.

For our own automatic answer typing approaches, our first observation is the benefit of ranking with $P(u|a)P(a|u)$ as opposed to $P(u|a)$ (Figure 1). Over most of the recall range, both the unexpanded (automatic) unit lists and the expanded unit lists improve in precision by a few percent when using both probabilistic scoring statistics. Secondly, note that both systems using the expanded unit lists can achieve almost 20% higher maximum recall than the unexpanded unit list systems. This provides strong justification for the small overhead of looking up

similar words for items on our unit list.

We next examine the AIPR performance of our comparison systems versus our best-performing automatic unit typing approach (Figure 2). The query-specific oracle is able to achieve the highest performance because of perfect knowledge of the units appropriate to a given question. However, its precision is only 42.2%. That is, the answer identification accuracy is limited because the correct answer shares its units with other numerical entities in the answer documents. Slightly worse, the adjective-specific oracle is limited to 34.2% precision. Unlike the query-specific oracle, if the question is "how long did WWII last?", the entities with the irrelevant units "meters" and "kilometers" must also be proposed as candidate answers because they occur in answers to other "how long" questions. This oracle thus provides a more appropriate upper bound on automatic unit-typing performance as our automatic approaches also build unit lists for adjectives rather than questions. Note again that unit lists for adjectives can be generated off-line whereas unit lists for specific questions need the query before processing.

In terms of recall, both upper-bound systems top out at around 78% (with our expanded systems reaching close to this at about 72%). At first, this number seems fairly disappointing: if how-adjective questions only have answer units in 78% of the cases, perhaps our typing approach is not entirely appropriate. On inspecting the actual misses, however, we find that 10 of the 16 missed questions correspond to "how old" questions. These are often answered without units (e.g. "at age 52."). Higher recall would be possible if the system defaults to extracting all numerical entities for "how old" questions. On the remaining questions, high recall can indeed be obtained.

Also of note is the clear disadvantage of using the standard fixed-category approach to how-question answer typing (Figure 2). Its precision runs at just under 5%, about a quarter of the lowest precision of any of our unit-list approaches at any recall value. However, fixed-category typing does achieve high recall, roughly 96%, missing only numerical entities unrecognized by Minipar. This high recall is possible because fixed-category typing does not miss answers for "how old" questions.

Both inferred approaches also perform worse than

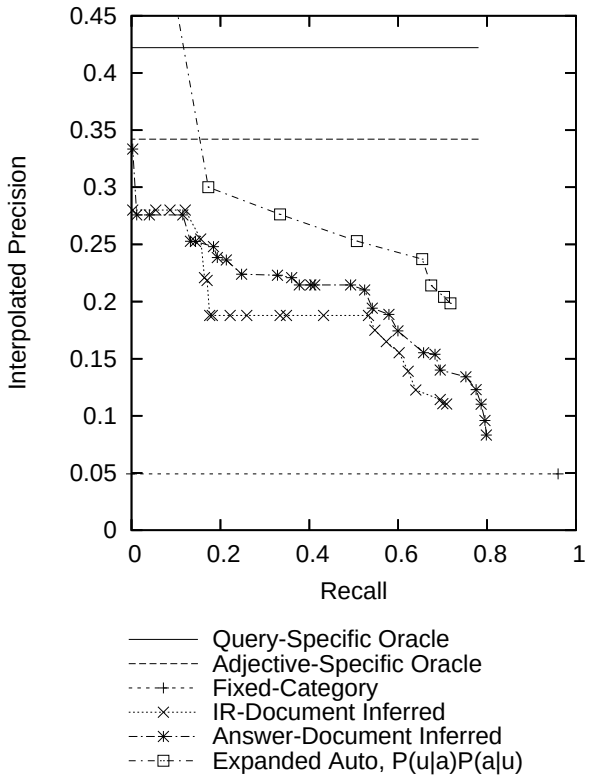

Figure 2: Microaveraged AIPR of our approach versus comparison systems.

our system (Figure 2). Thus inferring units from relevant documents does not seem promising, as even the unrealistic approach of inferring only from known answer documents cannot achieve as high in answer-identification precision. Also, realistically using IR-retrieved documents has universally lower AIPR. As expected, answer-document inferred recall plateaus at the same spot as the oracle systems, as it also requires a unit after each numerical entity (hurting it, again, on the "how old" questions). Despite their lower performance, note that these inferred approaches are completely orthogonal to our offline automatic answer-typing, so a future possibility remains to combine both kinds of systems within a unified framework.

7 Conclusions and Future Work

Although it is difficult to evaluate the impact of our approach until it is integrated into a full QA-system, we have clearly demonstrated the advantages of automatic answer typing for how-questions. We have

shown the improvements possible by ranking with different co-occurrence statistics, and the benefit of expanding unit lists with similar words. Experimental results show our approaches achieve superior AIPR performance over all realistic baselines.

In addition to proposing a competitive system, we believe we have established a framework and evaluation methodology that may be of use to other researchers. For example, although manual typing remains an option, our approach can at least provide a good set of candidate units to consider. Furthermore, a similar-word database can expand the list obtained by manual typing. Finally, users may also wish to rank the manual types in some way, and thus configure the system for a particular level of answer-identification precision/recall.

Our success with these unit lists has encouraged two main directions of future work. First, we plan to move to a discriminative approach to combining scores and weighting unit features using a small labeled set. Secondly, we will look at incorporating units into the information retrieval process. Our motivating example in Section 1 retrieved irrelevant documents when given to a search engine, and this seems to be a general trend in how-question IR. Less than 60% of the TREC how-questions have a unit of the correct type anywhere in the top ten documents returned by the PRISE IR engine, and less than half correspondingly had a correct answer in the top ten at all. Making the information retrieval process aware of the desired answer types will be an important future direction of QA research.

Acknowledgments

We gratefully acknowledge support from the Natural Sciences and Engineering Research Council of Canada, the Alberta Ingenuity Fund, and the Alberta Informatics Circle of Research Excellence.

References

E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. 2001. Data-Intensive Question Answering. In *TREC*.

S. Cucerzan and E. Agichtein. 2005. Factoid Question Answering over Unstructured and Structured Web Content. In *TREC*.

H. Daumé III and D. Marcu. 2006. Bayesian query-focused summarization. In *COLING-ACL*, pages 305–312.

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

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

A. Ittycheriah, M. Franz, W-J. Zhu, A. Ratnaparkhi, and R. Mammone. 2000. IBM's Statistical Question Answering System. In *TREC*.

Frank Keller and Mirella Lapata. 2003. Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484.

X. Li and D. Roth. 2002. Learning Question Classifi ers. In *COLING*, pages 556–562.

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. 1998a. Automatic retrieval and clustering of similar words. In *COLING-ACL*, pages 768–773.

D. Lin. 1998b. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems, First International Conference on Language Resources and Evaluation*.

G. Pass, A. Chowdhury, and C. Torgeson. 2006. A picture of search. In *The First International Conference on Scalable Information Systems*.

C. Pinchak and D. Lin. 2006. A Probabilistic Answer Type Model. In *EACL*.

J. Prager, J. Chu-Carroll, K. Czuba, C. Welty, A. Ittycheriah, and R. Mahindru. 2003. IBM's PIQUANT in TREC2003. In *TREC*.

D. Radev, H. Qi, Z. Zheng, S. Blair-Goldensohn, Z. Zhang, W. Fan, and J. Prager. 2001. Mining the Web for Answers to Natural Language Questions. In *CIKM*.

G. Salton and C. Buckley. 1988. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523.

E. Vorhees. 2002. Overview of the TREC 2002 question answering track. In *TREC*.

M. Wu, M. Duan, S. Shaikh, S. Small, and T. Strzalkowski. 2005. ILQUA – An IE-Driven Question Answering System. In *TREC*.