Automatic Question Answering Using the Web: Beyond the Factoid[*]

Radu Soricut
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292, USA
radu@isi.edu

Eric Brill
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
brill@microsoft.com

Abstract

In this paper we describe and evaluate a Question Answering (QA) system that goes beyond answering factoid questions. Our approach to QA assumes no restrictions on the type of questions that are handled, and no assumption that the answers to be provided are factoids. We present an unsupervised approach for collecting question and answer pairs from FAQ pages, which we use to collect a corpus of 1 million question/answer pairs from FAQ pages available on the Web. This corpus is used to train various statistical models employed by our QA system: a statistical chunker used to transform a natural language-posed question into a phrase-based query to be submitted for exact match to an off-the-shelf search engine; an answer/question translation model, used to assess the likelihood that a proposed answer is indeed an answer to the posed question; and an answer language model, used to assess the likelihood that a proposed answer is a well-formed answer. We evaluate our QA system in a modular fashion, by comparing the performance of baseline algorithms against our proposed algorithms for various modules in our QA system. The evaluation shows that our system achieves reasonable performance in terms of answer accuracy for a large variety of complex, non-factoid questions.

1 Introduction

The Question Answering (QA) task has received a great deal of attention from the Computational Linguistics and Information Retrieval research communities in the last few years (e.g., Text REtrieval Conference TREC 2001-2003). The definition of the task, however, is generally restricted to answering factoid questions: questions for which a complete answer can be given in 50 bytes or less, which is roughly a few words. Such a limitation reduces the difficulty of the question answering task, but imposes a great restriction in using the existent QA engines, because many of the questions people want answers for are *not* factoid questions. Questions of general and/or economic relevance, such as "How does a film qualify for an Academy Award?", or "Can I travel with the Ameripass in Mexico?" are beyond the scope and capabilities of most of the current QA engines.

It follows that there is a good economic incentive in moving the QA task to a more general level then the factoid: it is likely that a system able to answer complex questions of the type people generally and/or frequently ask has greater potential impact than one restricted to answering only factoid questions. A natural move is to recast the question answering task to handling questions people frequently ask or want answers for, as seen in Frequently Asked Questions (FAQ) lists.

In this paper, we make a first attempt towards solving a QA problem more generic than factoid QA, for which there are no restrictions on the type of questions that are handled, and there is no assumption that the answers to be provided are factoids. In our solution to this problem we exploit large document

[*] Journal of Information Retrieval - Special Issue on Web Information Retrieval, Volume ?, Number ??, 2006.
© 2006 Kluwer Academic Publishers

collections such as the Web for finding the answers (Agichtein et al., 2002; Kwok et al., 2001), and also employ learning mechanisms for answer extraction (Berger et al., 2000; Radev et al., 2001). We build our QA system around a noisy-channel architecture which exploits both a language model for answers and a transformation model for answer/question terms. The parameters of our models are trained on a corpus of 1 million question/answer pairs collected in an unsupervised manner from FAQ pages on the Web. Our evaluation shows that our system achieves reasonable performance in terms of answer accuracy for a large variety of complex, non-factoid questions.

The paper is organized as follows. The next section reviews the related work in Question Answering. Section 3 gives a high-level overview of the techniques used in this paper, and motivates their choice versus other techniques proposed to date. In Section 4 we describe the technique used to collect a large corpus of questions and answers from the Web, while Section 5 gives details about our QA system's architecture and the algorithms involved. In Section 6 we describe the evaluations we performed in order to assess our system's performance. In Section 7 we analyze some of the issues that negatively affected our system's performance.

2 Related Work

With very few exceptions, most of the work done in Question Answering focuses on answering factoid questions. Even with this limitation in place, factoid question answering is by no means an easy task. The challenges posed by answering factoid question have been addressed using a large variety of techniques. Question type determination has been widely used (Brill et al., 2001; Ittycheriah and Roukos, 2002; Hovy et al., 2001; Moldovan et al., 2002), and some of the QA typologies developed for this purpose (Hovy et al., 2001) contain as many as 80 nodes (which in many cases can be even further differentiated). Together with question type determination, question transformation and question expansion are also commonly used, via manually crafted question-to-query transformations (Brill et al., 2001), syntactic transformations of the question parse tree (Hovy et al., 2001; Moldovan et al., 2002), or WordNet exploitation for synonym and hypernym finding (Hovy et al., 2001; Pasca and Harabagiu, 2001; Prager et al., 2001).

Although initially these techniques have been employed in order to find answers on small and carefully constructed corpora, large collections of documents such as the Web have also been exploited using various answer-finding techniques. The wealth of information on the Web can be exploited for answer validation using redundancies (Brill et al., 2001), or directly for answer finding (Agichtein et al., 2002; Kwok et al., 2001). One of the most difficult challenges in automatic question answering has been recognized to be the lexical and stylistic gap between the question string and the answer string (Berger et al., 2000). Attempts for bridging this gap vary from the above mentioned question-to-query transformation techniques, to syntactically realized paraphrases (Hermjakob et al., 2002), to machine-learning techniques for question-to-answer transformation (Radev et al., 2001; Echihabi and Marcu, 2003) and answer-finding (Berger et al., 2000).

Most of the above mentioned techniques have been devised and evaluated in the context of factoid question answering. One of the few works that attempts to go beyond the factoid world is that of Agichtein et al. (2002), in which the authors focus on finding mechanisms for retrieving whole documents relevant for natural language-posed questions not restricted to factoid questions. The authors, however, do restrict the questions they use to evaluate their system to 4 general types of questions: "Who", "How", "Where", and "What". For each of these types of questions, they learn different transformations from question to more effective queries, which are then sent to a search engine to retrieve relevant documents from the Web. Although this technique has been shown effective for factoid as well as non-factoid questions, it still has problems handling complex questions, even when they do fall in the 4 types of questions handled. For example, a question such as, "How does a film qualify for an Academy Award?" requires an answer along the following lines: "A feature film must screen in a Los Angeles County theater in 35 or 70mm or in a 24-frame progressive scan digital format suitable for exhibiting in existing commercial digital cinema sites for paid admission for seven consecutive days. The seven day run must begin before midnight, December 31, of the qualifying year. […]"; it is extremely unlikely that any type of question reformulation will increase the chance of finding the document containing the answer.

The work that is most closely related to the present work is the work of Berger et al. (2000), in which the authors explore several statistical techniques for answer finding using as a data source FAQ documents. Their work, however, does not describe an end-to-end QA system, but rather explores and compares several techniques for finding the right answers given documents known to contain the answers. The conclusion of Berger et al. (2000), which we support and extend in the present work, is that the lexical chasm between question and answer terms can be successfully bridged using techniques inspired from Statistical Machine Translation (Brown et al., 1993). Such techniques have also been exploited for answering factoid questions by Radev et al. (2001), who propose "translating" questions into answers and then perform answer finding based on the "translations" found, and Echihabi and Marcu (2003), who describe a noisy-chanel approach to factoid question-answering, in which the similarity between the factoid answers and the posed questions are computed using noisy channel transformations.

3 Beyond Factoid Question Answering

The techniques we propose for handling our extended QA task are mainly statistically driven. During the early stages of our system development, we have observed that question reformulations (such as paraphrasing, or direct "translation" of question terms into answer terms) are less likely to have a positive impact when dealing with non-factoid questions, presumably due to the large space of possible ways of expressing the answers. On the other hand, we have observed (and the evaluation results presented in Section 6 confirmed) that the recall of retrieving the appropriate documents from the Web increases significantly if the question terms are presented to the search engine for exact matching of phrases, rather than as bag-of-words, provided an appropriate segmentation of the question into phrases can be found. Using this observation, we propose as "question transformation" a chunking technique based on co-occurrence statistics.

After submitting the obtained chunks for exact matching to a search engine, we retrieve the *N* most relevant documents from the Web, and search for the best answer we can find in the retrieved documents. Techniques such as tf·idf have been previously shown (Berger et al., 2000) to perform poorly for answer extraction, precisely because it is difficult to find good answers when only considering their similarity with question terms. In order to bridge this lexical gap between question and answer terms, we employ a statistical model to assess the likelihood that a proposed answer is indeed an answer to the posed question. Such a model can be seen as a "translation model" between answers and questions, and the quantity $p(q|a)$ is used to assess the relevance of answer *a* to question *q*. For example, the answer terms Mexico, Mexicans, PassPort, Boje, Tijuana, Reynosa, Laredo, CanAm, Ameripass will all have reasonably high probability of translating into the question term Mexico.

Finding a best answer does not mean finding only the content words that might be relevant to a given question. A bunch of title words describing the content of a document might have many relevant words needed to answer a question, but would not qualify as the best answer that can be given. It follows that in finding the best answer we also need to assess the likelihood that the proposed answer is a well-formed answer, which can be achieved using an "answer language model". The model is used to compute the quantity $p(a)$, which can be seen as the likelihood that a proposed answer *a* resembles the answers seen in the training corpus in terms of length, coherence, etc.

4 A Question-Answer Corpus of FAQs

In order to train the statistical models described in the previous section, we first need to build a large training corpus consisting of question-answer pairs of a broad lexical coverage. Previous work using FAQs as a source for finding an appropriate answer (Burke et al., 1997) or for learning lexical correlations (Berger et al., 2000) focused on using the publicly available Usenet FAQ collection and other non-public FAQ collections, and reportedly worked with an order of thousands of question-answer pairs.

Our approach to question/answer pair collection takes a different path. If one poses the simple query "FAQ" to a search engine, one can observe that roughly 85% of the returned URL strings corresponding to genuine FAQ pages contain the substring "faq", while virtually all of the URLs that contain the substring "faq" are genuine FAQ pages. It follows that, if one has access to a large collection of the Web's existent URLs, a simple pattern-matching for "faq" on these URLs will have a recall close to 85% and precision close to 100% on returning FAQ URLs from those available in the collection. For the purposes of the present research, we have used Microsoft Research's proprietary URL collection of approximately 1 billion URLs, and using this technique we extracted roughly 2.7 million URLs containing the (uncased) string "faq", which amounts to roughly 2.3 million FAQ URLs to be used for collecting question/answer pairs.

The collected FAQ pages displayed a variety of formats and presentations. It seems that the variety of ways questions and answers are usually listed in FAQ pages does not allow for a simple high-precision high-recall solution for extracting question/answer pairs: if one assumes that only certain templates are used when presenting FAQ lists, one can obtain clean question/answer pairs at the cost of losing many other such pairs (which happen to be presented in different templates); on the other hand, assuming very loose constraints on the way information is presented on such pages, one can obtain a bountiful set of question/answer pairs, plus other pairs that do not qualify as such. We settled for a two-step approach: a first recall-oriented pass based on universal indicators such as end-of-sentence punctuation (".", "!", "?") and hypertext cues ("<P>", "<BR>", "<HR>", etc.) allowed us to retrieve most of the question/answer pairs, along with other noise data. A second precision-oriented pass used several filters to reduce the level of noise of the question/answer pair corpus; these filters were language identification (other than English questions and answers were not considered), lexical cues (start-of-question cues such as "Can", "How", "Is", "Should", "What", etc. – 31 in total), and length constrains (after a question was identified, only the first 3 following sentences were considered as part of the answer; the remaining sentences, if any, were ignored until the next question was found). Using this method, we were able to collect a total of roughly 1 million question/answer pairs, exceeding by 1-2 orders of magnitude the amount of data previously used for learning statistics on questions and answers.

5 A QA System Architecture

The architecure of our QA system is presented in Figure 1. There are 4 separate modules that handle various stages in the system's pipeline: the first module is called Question2Query, in which questions posed in natural language are transformed into phrase-based queries before being handed down to the SearchEngine module. The second module is an Information Retrieval engine which takes a query as input and returns a list of documents deemed to be relevant to the query in a sorted manner. A third module, called Filter, is in charge of filtering out the returned list of documents, in order to provide acceptable input to the next module. The forth module, AnswerExtraction, analyzes the content presented and chooses the text fragment deemed to be the best answer to the posed question.

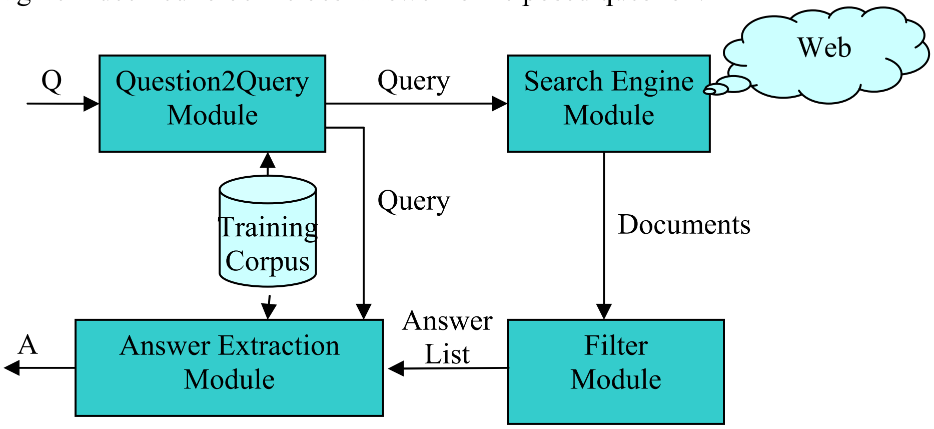

Figure 1: The QA system architecture

This architecture allows us to flexibly test for various changes in the pipeline and evaluate their overall effect. We present next detailed descriptions of how each module works, and outline several choices for the algorithms implementing these modules that present themselves as acceptable options to be evaluated.

5.1 The Question2Query Module

A query is defined to be a keyword-based string that users are expected to feed as input to a search engine. Such a string is often thought of as a representation for a user's "information need", and being proficient in expressing one's "need" in such terms is one of the key points in successfully using a search engine. A natural language-posed question can be thought of as such a query. It has the advantage that it forces the user to pay more attention to formulating the "information need" (and not typing the first keywords that come to mind). It has the disadvantage that it contains not only the keywords a search engine normally expects, but also a lot of extraneous "details" as part of its syntactic and discourse constraints, plus an inherently underspecified unit-segmentation problem, which can all confuse the search engine.

To counterbalance some of these disadvantages, we build a statistical chunker that uses a dynamic programming algorithm to chunk the question into chunks/phrases. The chunker is trained on the answer side of the Training corpus in order to learn 2 and 3-word collocations, defined using the likelihood ratio of Dunning (1993). Note that we are chunking the question using answer-side statistics, precisely as a measure for bridging the stylistic gap between questions and answers. In Figure 2 we list the top 10 ranked trigram collocations found in our Training corpus using the likelihood ratio statistics (the first column is the likelihood ratio score; the second column is the number of counts in the corpus). Strong collocations such as "the United States" and "letter of credit" receive high likelihood ratio scores even if their raw count is 3 times less than the raw counts of other weaker collocations, such as "If you are". Because our Training corpus is collected in an unsupervised manner, some inherent noise exists in the collocation table as well: the French collocation "Date de publication" receives a high score because it is a frequently seen trigram in the style of text we collected, and also because the word "de" was rarely seen in other contexts (which is, according to the likelihood ratio criterion, a good indication of collocation with its neighboring words).

2585.1	266	: Re :
2244.5	186	For example ,
2041.9	370	If you have
1944.5	109	Date de publication
1772.8	108	the United States
1746.6	96	re r vision
1729.4	109	de publication :
1709.5	109	letter of credit
1641.2	170	There is no
1595.3	365	If you are

Figure 2. Top 10 ranked trigram collocations extracted from the Training corpus of answers using the likelihood ratio statistics

Our statistical chunker uses the extracted collocation statistics to make an optimal chunking using a Dijkstra-style dynamic programming algorithm. For a given input question from which the final punctuation mark is removed, our algorithm assigns a weight of 1 for each unigram, and a weight equal to the likelihood ratio for each bigram and trigram found in the input question that has a likelihood ratio greater than 1 as computed from the Training corpus. A maximum scoring path is found in the input question by a dynamic programming search. If the maximum scoring path contains unigrams which are found in our stop-word list, those unigrams are not output by the chunker. In Figure 3 we present two examples of the results returned by our statistical chunker. Important cues such as “differ from”, “herbal medications”, “tap water”, “to drink” are presented as phrases to the search engine, therefore increasing the recall of the search when considering the top *N* returned pages; stop-words such as “if” and “my” (from the second example) do not appear in the segmented question. Note that, unlike a segmentation offered by a parser (Hermjakob et al., 2002), our phrases are not necessarily syntactic constituents. A statistics-based chunker also has the advantage that it can be used “as-is” for question segmentation in languages other than English, provided training data (i.e., plain written text) is available.

How do herbal medications differ from conventional drugs?
"How do" "herbal medications" "differ from" "conventional" "drugs"

How can I find out if my tap water is safe to drink?
“How can” “I” “find out” “tap water” “is safe” “to drink”

Figure 3: Question segmentation into query using a statistical chunker

5.2 The SearchEngine Module

This module consists of a configurable interface with available off-the-shelf search engines. It currently supports MSNSearch, http://search.msn.com (visited August 10th, 2003), and Google, http://www.google.com (visited August 10th, 2003). Switching from one search engine to another allowed us to measure the impact of the IR engine on the QA task.

5.3 The Filter Module

This module is in charge of providing the AnswerExtraction module with the content of the pages returned by the search engine, after three filtering steps. The first step is to reduce the volume of pages returned to only a manageable amount. We implement this step as choosing to return the first *N* hits provided by the search engine. The other two filtering steps performed by the Filter Module are tokenization and segmentation of text into sentences.

One more filtering step was needed for evaluation purposes only: because both our training and test data were collected from the Web (using the technique described in Section 4), there was a good chance that asking a question previously collected returned its already available answer, thus optimistically biasing our evaluation. The Filter Module therefore had access to the reference answers for the test questions as well, and ensured that, if the reference answer matched a string in some retrieved page, that page was discarded. Moreover, we found that slight variations of the same answer could defeat the purpose of the string-matching check. For the purpose of our evaluation, we considered that if the question/reference answer pair had a string of 10 words or more identical with a string in some retrieved page, that page was discarded as well. Note that, outside the evaluation procedure, the string-matching filtering step is not needed, and our system’s performance can only increase by removing it.

5.4 The AnswerExtraction Module

Authors of previous work on statistical approaches to answer finding (Berger et al., 2000) emphasized the need to “bridge the lexical chasm” between the question terms and the answer terms. Berger et al. suggested that techniques that did not bridge the lexical chasm, such as tf·idf, were likely to perform worse than techniques that did.

For comparison purposes, we consider two different algorithms for our AnswerExtraction module: a new extraction algorithm that does not bridge the lexical chasm, based on N-gram co-occurrences between the question terms and the answer terms, to be used as baseline; and an extraction algorithm that bridges the lexical chasm using Statistical Machine Translation inspired techniques (Brown et al., 1993) in order to find the best answer for a given question.

For both algorithms, each 3 consecutive sentences from the documents provided by the Filter module form a potential answer. The choice of 3 sentences comes from the average number of sentences in the answers from our training corpus. The choice of consecutiveness comes from the empirical observation that answers built up from consecutive sentences tend to be more coherent and contain more non-redundant information than answers built up from non-consecutive sentences.

5.4.1 N-gram Co-Occurrence Statistics for Answer Extraction

N-gram co-occurrence statistics have been successfully used in automatic evaluation (Papineni et al. 2002, Lin and Hovy 2003), and more recently as training criteria in statistical machine translation (Och 2003). For answer extraction, we implemented an algorithm based on the BLEU score of Papineni et al. (2002) as a means of assessing the overlap between the question and the proposed answers.

As opposed to the BLEU score, which computes an overlap score over an entire collection of proposed answers, our algorithm computes an overlap score for each proposed answer. For each individual candidate answer *C*, we define *S(C,n)* as the multi-set of *n*-grams obtained from *C*. For a fixed *n*, we define an *n*-gram overlap score as:

$$P(n) = \frac{\sum_{ngram \in S(C,n)} Count_{clip}(ngram)}{\sum_{ngram \in S(C,n)} Count(ngram)} \quad (1)$$

where *Count(ngram)* is the number of *n*-gram counts, and $Count_{clip}(ngram)$ is the maximum number of co-occurences of *ngram* in the candidate answer and the given question. Because the denominator in the *P(n)* formula consists of a sum over the *n*-grams of the proposed candidate answer, this formula is essentially a precision-oriented formula, which penalizes verbose candidate answers by default. This precision score, however, gets artificially higher for shorter and shorter candidate answers. This is offset by adding a brevity penalty, *BP*:

$$BP = \begin{cases} 1, & if \quad B \cdot |c| \geq |r| \\ e^{(1-|r|/B|c|)}, & if \quad B \cdot |c| < |r| \end{cases} \quad (2)$$

where |*c*| is the length of the proposed answer, |*r*| is the length of the question, and *B* is a brevity constant. An overlap score is defined now as:

$$NG = BP \cdot \exp\left(\sum_{n=1}^{4} w_n \log(P(n))\right) \quad (3)$$

This score can be interpreted as a weighted linear average of *n*-gram overlap scores for unigrams, bigrams, trigrams, and four-grams. As the values of the *n*-gram overlap scores decrease roughly exponentially with the increase of *n*, the logarithm is needed to obtain a linear average. If the score *P(n)* goes to zero for some $n \leq 4$, the term is excluded from the sum.

For each potential answer, the overlap with the question was assessed with *NG* with the brevity constant *B* set to 3; this way, answers shorter than 3 times the length of the question will be penalized via the brevity penalty. The best scoring potential answer was presented by the AnswerExtraction Module as the answer to the question.

5.4.2 Statistical Translation for Answer Extraction

As proposed by Berger et al. (2000), the lexical gap between questions and answers can be bridged by a statistical translation model between answer terms and question terms. Their model, however, uses only an Answer/Question translation model (see Figure 4) as a means to find the answer.

A more complete model for answer extraction can be formulated in terms of a noisy channel, along the lines of Berger and Lafferty (1999) for the Information Retrieval task, as illustrated in Figure 4: an answer generation model proposes an answer *A* according to an answer generation probability distribution; answer *A* is further transformed into question *Q* by an answer/question translation model according to a question-given-answer conditional probability distribution. The task of the AnswerExtraction algorithm is to take the given question *q* and find an answer *a* in the potential answer list that is most likely both an appropriate and well-formed answer.

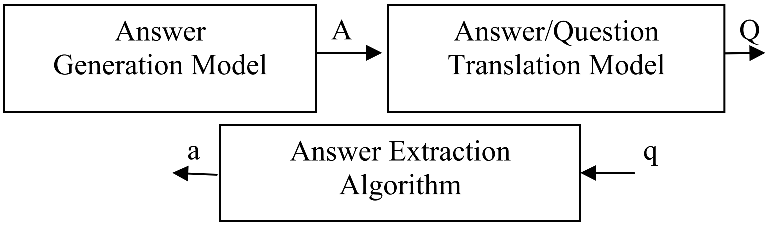
Answer Generation Model
A
Answer/Question Translation Model
Q
a
Answer Extraction Algorithm
q

Figure 4: A noisy-channel model for answer extraction

The AnswerExtraction procedure employed depends on the task *T* we want it to accomplish. Let the task *T* be defined as "find a 3-sentence answer for a given question". Then we can formulate the algorithm as finding the *a-posteriori* most likely answer given question and task, and write it as $p(a|q,T)$. We can use Bayes' law to write this as:

$$p(a \mid q,T) = \frac{p(q \mid a,T) \cdot p(a \mid T)}{p(q \mid T)} \tag{4}$$

Because the denominator is fixed given question and task, we can ignore it and find the answer that maximizes the probability of being both a well-formed and an appropriate answer as:

$$a = \arg\max_{a} \underbrace{p(a \mid T)}_{question-independent} \cdot \underbrace{p(q \mid a,T)}_{question-dependent} \tag{5}$$

The decomposition of the formula into a question-independent term and a question-dependent term allows us to separately model the quality of a proposed answer *a* with respect to task *T*, and to determine the appropriateness of the proposed answer *a* with respect to question *q* to be answered in the context of task *T*.

Because task *T* fits the characteristics of the question-answer pair corpus described in Section 4, we can use the answer side of this corpus to compute the prior probability $p(a|T)$. The role of the prior is to help downgrading those answers that are too long or too short, or are otherwise not well-formed. We use a standard trigram language model to compute the probability distribution $p(\cdot|T)$.

The mapping of answer terms to question terms is modeled using Brown et al.'s (1993) simplest model, called IBM Model 1. For this reason, we call our model Model 1 as well. Under this model, a question is generated from an answer *a* of length *n* according to the following steps: first, a length *m* is chosen for the question, according to the distribution $\psi(m|n)$ (we assume this distribution is uniform); then, for each position *j* in *q*, a position *i* in *a* is chosen from which q_j is generated, according to the distribution $t(\cdot| a_i)$. The answer is assumed to include a *NULL* word, whose purpose is to generate the content-free words in the question (such as in "Can you please tell me…?"). The correspondence between the answer terms and the question terms is called an alignment, and the probability $p(q|a)$ is computed as the sum over all possible alignments. We express this probability using the following formula:

$$p(q \mid a) = \psi(m \mid n) \prod_{j=1}^{m} \left(\frac{n}{n+1} \left(\sum_{i=1}^{n} t(q_j \mid a_i) \cdot c(a_i \mid a) \right) + \frac{1}{n+1} t(q_j \mid NULL) \right) \tag{6}$$

where $t(q_j| a_i)$ are the probabilities of "translating" answer terms into question terms, and $c(a_i|a)$ are the relative counts of the answer terms. Our parallel corpus of questions and answers can be used to compute the translation table $t(q_j| a_i)$ using the EM algorithm, as described by Brown et al. (1993). Note that, similarly with the statistical machine translation framework, we deal here with "inverse" probabilities, i.e. the probability of a question term given an answer, and not the more intuitive probability of answer term given question.

Following Berger and Lafferty (1999), an even simpler model than Model 1 can be devised by skewing the translation distribution *t(·| a_i)* such that all the probability mass goes to the term a_i. This simpler model is called Model 0. In Section 6 we evaluate the proficiency of both Model 1 and Model 0 in the answer extraction task.

6 Evaluations and Discussions

We evaluated our QA system systematically for each module, in order to assess the impact of various algorithms on the overall performance of the system. The evaluation was done by a human judge on a set of 115 Test questions, which contained a large variety of non-factoid questions. Each answer was rated as either *correct*(C), *somehow related*(S), *wrong*(W), or *cannot tell*(N). The *somehow related* option allowed the judge to indicate the fact that the answer was only partially correct (for example, because of missing information, or because the answer was more general/specific than required by the question, etc.). The *cannot tell* option was used in those cases when the validity of the answer could not be assessed. Note that the judge did not have access to any reference answers in order to assess the quality of a proposed answer. Only general knowledge and human judgment were involved when assessing the validity of the proposed answers. Also note that, mainly because our system's answers were restricted to a maximum of 3 sentences, the evaluation guidelines stated that answers that contained the right information plus other extraneous information were to be rated *correct*. In Section 7 we list examples of answers rated *correct*, *somehow related, wrong*, and *cannot tell* to help better understand our evaluation and to discuss some of the most frequent errors made by our system.

For the answer sets obtained by different systems on the Test questions, we estimated the performance of these systems using the formula (|C|+0.5|S|)/(|C|+|S|+|W|). This formula gives a score of 1 if the answers that are not "N" rated are all considered *correct*, and a score of 0 if they are all considered *wrong*. A score of 0.5 means that, on average, 1 out of 2 questions is answered correctly.

6.1 Question2Query Module Evaluation

We evaluated the Question2Query module while keeping fixed the configuration of the other modules (MSNSearch as the search engine, the top 10 hits in the Filter module), except for the AnswerExtraction module, for which we tested both the N-gram co-occurrence based algorithm (NG-AE) and a Model 1 based algorithm (M1e-AE, see Section 6.4).

The evaluation assessed the impact of the statistical chunker used to transform questions into queries, against the baseline strategy of submitting the question as-is to the search engine. As illustrated in Figure 5, the overall performance of the QA system significantly increased when the question was segmented before being submitted to the SearchEngine module, for both AnswerExtraction algorithms. The score increased from 0.18 to 0.23 when using the NG-AE algorithm and from 0.34 to 0.38 when using the M1e-AE algorithm.

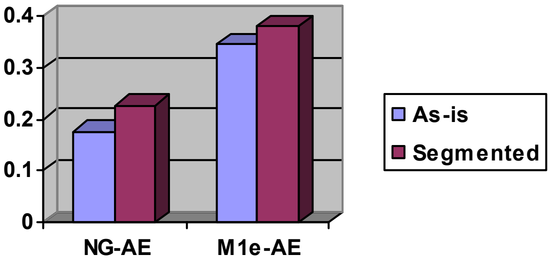

Figure 5: Evaluation of the Question2Query module

6.2 SearchEngine Module Evaluation

The evaluation of the SearchEngine module assessed the impact of different search engines on the overall system performance. We fixed the configurations of the other modules (segmented question for the Question2Query module, top 10 hits in the Filter module), except for the AnswerExtraction module, for which we tested the performance while using for answer extraction the NG-AE, M1e-AE, and ONG-AE algorithms. The later algorithm works exactly like NG-AE, with the exception that the potential answers are compared with a reference answer available to an Oracle, rather than against the question. The performance obtained using the ONG-AE algorithm can be thought of as indicative of the ceiling in the performance that can be achieved by an AE algorithm given the potential answers available.

As illustrated in Figure 6, both the MSNSearch and Google search engines achieved comparable performance accuracy. The scores were 0.23 and 0.24 when using the NG-AE algorithm, 0.38 and 0.37 when using the M1e-AE algorithm, and 0.46 and 0.46 when using the ONG-AE algorithm, for MSNSearch and Google, respectively. As a side note, it is worth mentioning that only 5% of the URLs returned by the two search engines for the entire Test set of questions overlapped. Therefore, the comparable performance accuracy was not due to the fact that the AnswerExtraction module had access to the same set of potential answers, but rather to the fact that the 10 best hits of both search engines provide similar answering options.

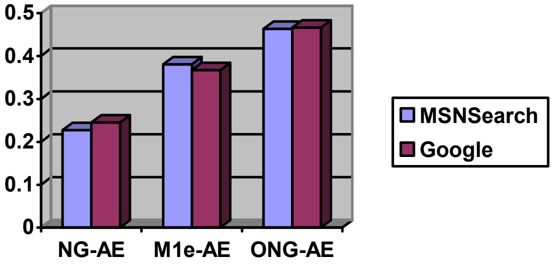

Figure 6: MSNSearch and Google give similar performance both in terms of realistic AE algorithms and oracle-based AE algorithms

6.3 Filter Module Evaluation

As mentioned in Section 5, the Filter module filters out the low score documents returned by the search engine and provides a set of potential answers extracted from the N-best list of documents. The evaluation of the Filter module therefore assessed the trade-off between computation time and accuracy of the overall system: the size of the set of potential answers directly influences the accuracy of the system while increasing the computation time of the AnswerExtraction module. The ONG-AE algorithm gives an accurate estimate of the performance ceiling induced by the set of potential answers available to the AnswerExtraction Module.

As seen in Figure 7, there is a significant performance ceiling increase from considering only the document returned as the first hit (0.36) to considering the first 10 hits (0.46). There is only a slight increase in performance ceiling from considering the first 10 hits to considering the first 50 hits (0.46 to 0.49).

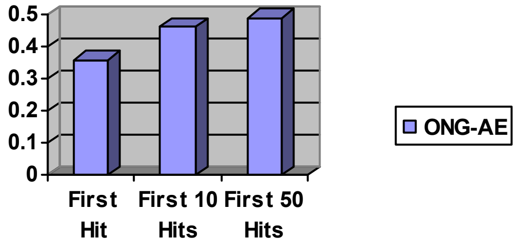

Figure 7: The scores obtained using the ONG-AE answer extraction algorithm for various N-best lists

6.4 AnswerExtraction Module Evaluation

The Answer-Extraction module was evaluated while fixing all the other module configurations (segmented question for the Question2Query module, MSNSearch as the search engine, and top 10 hits in the Filter module). The algorithm based on the BLEU score, NG-AE, and its Oracle-informed variant ONG-AE, do not depend on the amount of training data available, and therefore they performed uniformly at 0.23 and 0.46, respectively (Figure 8). The score of 0.46 can be interpreted as a performance ceiling of the AE algorithms given the available set of potential answers.

The algorithms based on the noisy-channel architecture displayed increased performance with the increase in the amount of available training data, reaching as high as 0.38. An interesting observation is that the extraction algorithm using Model 1 (M1-AE) performed more poorly than the extraction algorithm using Model 0 (M0-AE), for the available training data. Our explanation is that the probability distribution of question terms given answer terms learnt by Model 1 is well informed (many mappings are allowed) but badly distributed, whereas the probability distribution learnt by Model 0 is poorly informed (indeed, only one mapping is allowed), but better distributed. Note the steep learning curve of M1-AE, whose performance gets increasingly better as the distribution probabilities of various answer terms (including the NULL word) become more informed (more mappings are learnt), compared to the gentle learning curve of M0-AE, whose performance increases slightly only as more words become known as self-translations to the system (and the distribution of the NULL word gets better approximated).

From the above analysis, it follows that a model whose probability distribution of question terms given answer terms is both well informed and well distributed is likely to outperform both M1-AE and M0-AE. Such a model was obtained when Model 1 was trained on both the question/answer parallel corpus from Section 4 and an artificially created parallel corpus in which each question had itself as its "translation". This training regime allowed the model to assign high probabilities to identity mappings (and therefore be better distributed), while also distributing some probability mass to other question-answer term pairs (and therefore be well informed). We call the extraction algorithm that uses this model M1e-AE, and the top score of 0.38 was obtained by M1e-AE when trained on 1 million question/answer pairs. Note that the learning curve of algorithm M1e-AE in Figure 8 indeed indicates that this answer extraction procedure is well informed about the distribution probabilities of various answer terms (it has the same steepness in the learning curve as for M1-AE), while at the same time uses a better distribution of the probability mass for each answer term compared to M1-AE (it outperforms M1-AE by roughly a constant amount for each training set size in the evaluation).

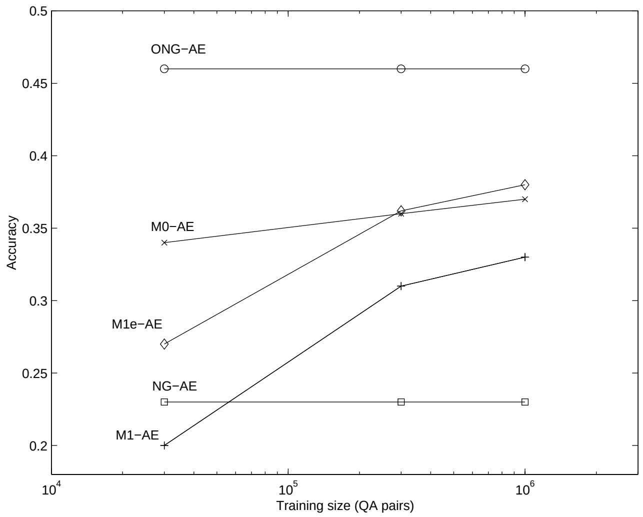

Figure 8: The performance of our QA system with various answer extraction algorithms and different training set sizes

7 Performance issues

In building our system, we have demonstrated that a statistical model can capitalize on large amounts of readily available training data to achieve reasonable performance on answering non-factoid questions. Our system was able to successfully answer a large variety of complex, non-factoid questions; here are some of the answers that were rated *correct*:

Q: Can I travel with the Ameripass in Mexico?
A: The Ameripass includes all of USA excluding Alaska. In Canada, it is valid for travel to Montreal, Quebec, Toronto, Ontario, Vancouver BC, and New Westminster BC. In Mexico it is valid for travel to Matamoros, Nuevo Laredo, Tamaulipas and Tijuana Baja California.

Q: What is a PIC Freeze?
A: How to Prevent Slamming: Institute a PIC Freeze The best way to keep from being slammed, or having a long distance company change your service without your permission, is to request a "PIC freeze" for your line. PIC, or pre-subscribed interexchange carrier, is the telecom industry's term for a long distance company.

The answers that were rated *somehow related* usually fall in one of the following categories:

- The answer is referring to either a smaller or a larger class of entities than the one asked for in the question (here we list an example where the answer involves a smaller class):
  Q: Can I drink alcohol while taking antibiotics?
  A: Be careful drinking alcohol when you are taking Augmentin tablets. Drinking alcohol while taking these tablets does not usually cause any problems. However, some people who drink alcohol while taking antibiotics similar to Augmentin tablets have experienced adverse effects.
- The answer is addressing the question only indirectly, but nevertheless gives useful information:
  Q: Is there any way to get rid of the undesirable grasses in the lawn?
  A: Water only when the grass starts to show symptoms of drought. Irrigating deeply and less often is much better than frequent, shallow watering. Light, frequent watering will result in short root systems and only encourage undesirable grasses to invade the turf.

For those questions which received answers rated *wrong*, we identified some of the most frequent causes which led to erroneous answers:

- The answer was not in the retrieved pages (see the 46% performance ceiling given by the Oracle)
- The answer was of the wrong "type" (e.g., an answer for "how-to" instead of "what-is"):
  Q: What are best graduate schools for AI?
  A: If you are applying for grad school in AI, and you did some research for an AI person, even if it was long ago and you don't consider it to be relevant, get a recommendation if you think that the person likes you at all. […]
- The answer pointed to where an answer might be instead of answering the question
  Q: What do research studies say about massage therapy?
  A: It supports research on the efficacy of therapeutic massage through the public, charitable AMTA Foundation. Additional information about massage therapy and about AMTA is available via the Web at www.amtamassage.org.
- The translation model overweighed the answer language model (too good a "translation", too bad an answer)
  Q: What are private and public keys?
  A: Private and public keys Private and public keys Algorithms can use two types of keys: private and public.
- It did not pick up the key content word (in the example below, eggs)
  Q: What makes eggs have thin, brittle shells?
  A: The soft-shelled clams, such as steamer, razor, and geoduck clams, have thin brittle shells that can't completely close. Cod - A popular lean, firm, white meat fish from the Pacific and the North Atlantic.

It is worth pointing out that most of these errors do not arise from within a single module, but rather they are the result of various interactions between modules that miss on some relevant information.

For the sake of completeness, we also list here an answer that was rated *cannot tell* (in this case, because it was not clear whether "this medicine" refers to an antibiotic or not):
Q: Can I drink alcohol while taking antibiotics?
A: People should not drink alcohol while taking this medicine because the combination may cause severe abdominal distress and vomiting.

8 Conclusions

Previous work on question answering has focused almost exclusively on building systems for handling factoid questions. These systems have recently achieved impressive performance (Moldovan et al., 2002). The world beyond the factoid questions, however, is largely unexplored, with few notable exceptions (Berger et al., 2000; Agichtein et al., 2002; Girju 2003). The present paper attempts to explore the portion related to answering FAQ-like questions, without restricting the domain or type of the questions to be handled, or restricting the type of answers to be provided. While we still have a long way to go in order to achieve robust non-factoid QA, this work is a step in a direction that goes beyond restricted questions and answers.

We consider the present QA system as a baseline on which more finely tuned QA architectures can be built. Learning from the experience of factoid question answering, one of the most important features to be added is a question typology for the FAQ domain. Efforts towards handling specific question types, such as causal questions, are already under way (Girju 2003). A carefully devised typology, correlated with a systematic approach to fine tuning, seem to be the lessons for success in answering both factoid and beyond factoid questions.

References

Eugene Agichtein, Steve Lawrence, and Luis Gravano. 2002. Learning to Find Answers to Questions on the Web. In ACM Transactions on Internet Technology, 4(2): 129—162.

Adam L. Berger, John D. Lafferty. 1999. Information Retrieval as Statistical Translation. In Proceedings of 1999 ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99), Berkeley, CA, August 1999, pp. 222—229.

Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, Vibhu Mittal. 2000. Bridging the Lexical Chasm: Statistical Approaches to Answer-Finding. In Proceedings of the 23rd Annual Conference on Research and Development in Information Retrieval (ACM SIGIR), Athens, Greece, pp 192—199.

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, Andrew Ng. 2001. Data-Intensive Question Answering. In Proceedings of the 10th Text REtrieval Conference(TREC-10), Gaithersburg, MD, November 2001.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics, 19(2):263—312.

Robin Burke, Kristian Hammond, Vladimir Kulyukin, Steven Lytinen, Noriko Tomuro, and Scott Schoenberg. 1997. Question Answering from Frequently-Asked Question Files: Experiences with the FAQ Finder System. AI Magazine, 18(2): 57—66.

Ted Dunning. 1993. Accurate Methods for the Statistics of Surprise and Coincidence. Computational Linguistics, 19(1):61—74.

Abdessamad Echihabi and Daniel Marcu. 2003. A Noisy-Channel Approach to Question Answering. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03), Sapporo, Japan, July 7-12.

Roxana Girju. 2003. Automatic Detection of Causal Relations for Question Answering. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003), Workshop on "Multilingual Summarization and Question Answering - Machine Learning and Beyond", Sapporo, Japan, July 7-12.

Ulf Hermjakob, Abdessamad Echihabi, and Daniel Marcu. 2002. Natural Language Based Reformulation Resource and Web Exploitation for Question Answering. In Proceedings of the 11th Text REtrieval Conference(TREC-11), Gaithersburg, MD.

Eduard H. Hovy, Ulf Hermjakob, and Chin-Yew Lin. 2001. The Use of External Knowledge in Factoid QA. In Proceedings of the 10th Text REtrieval Conference(TREC-10), Gaithersburg, MD, November 2001.

Abraham Ittycheriah and Salim Roukos. 2002. IBM's Statistical Question Answering System-TREC 11. In Proceedings of the 11th Text REtrieval Conference(TREC-11), Gaithersburg, MD.

Cody C. T. Kwok, Oren Etzioni, Daniel S. Weld. 2001. Scaling Question Answering to the Web. ACM Transactions on Information Systems, 19(3):242—262.

Chin-Yew Lin and E.H. Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In Proceedings of 2003 Human Language Technology Conference (HLT-NAACL 2003), Edmonton, Canada, May 27 - June 1, pp 150—157.

Dan Moldovan, Sanda Harabagiu, Roxana Girju, Paul Morarescu, Finley Lacatusu, Adrian Novischi, Adriana Badulescu, Orest Bolohan. 2002. LCC Tools for Question Answering. In Proceedings of the 11th Text REtrieval Conference(TREC-11), Gaithersburg, MD.

Franz Joseph Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03), Sapporo, Japan, July 7-12.

Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02), Philadephia, PA, July 7-12, pp 311—318.

Marius Pasca, Sanda Harabagiu, 2001. The Informative Role of WordNet in Open-Domain Question Answering. In Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001), Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations, Pittsburgh, Pennsilvania, June 2001.

John M. Prager, Jennifer Chu-Carroll, Krysztof Czuba. 2001. Use of WordNet Hypernyms for Answering What-Is Questions. In Proceedings of the 10th Text REtrieval Conference(TREC-10), Gaithersburg, MD, November 2001.

Dragomir Radev, Hong Qi, Zhiping Zheng, Sasha Blair-Goldensohn, Zhu Zhang, Weiguo Fan, and John Prager. 2001. Mining the Web for Answers to Natural Language Questions. In Proceedings of the 10th International Conference on Information and Knowledge Management, Atlanta, GA, pp 143—150.