Automatic Question Answering: 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 system that goes beyond answering factoid questions. We focus on FAQ-like questions and answers, and build our system around a noisy-channel architecture which exploits both a language model for answers and a transformation model for answer/question terms, trained on a corpus of 1 million question/answer pairs collected from the Web.

1 Introduction

The Question Answering (QA) task has received a great deal of attention from the Computational Linguistics research community 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. 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, such as question parsing (Hovy et al., 2001; Moldovan et al., 2002), question-type determination (Brill et al., 2001; Ittycheraih and Roukos, 2002; Hovy et al., 2001; Moldovan et al., 2002), WordNet exploitation (Hovy et al., 2001; Pasca and Harabagiu, 2001; Prager et al., 2001), Web exploitation (Brill et al., 2001; Kwok et al., 2001), noisy-channel transformations (Echihabi and Marcu, 2003), semantic analysis (Xu et al., 2002; Hovy et al., 2001; Moldovan et al., 2002), and inferencing (Moldovan et al., 2002).

The obvious limitation of any factoid QA system is that many questions that people want answers for are *not* factoid questions. It is also frequently the case that non-factoid questions are the ones for which answers cannot as readily be found by simply using a good search engine. It follows that there is a good economic incentive in moving the QA task to a more general level: 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. These questions are sometimes factoid questions (such as, "What is Scotland's national costume?"), but in general are more complex questions (such as, "How does a film qualify for an Academy Award?", which 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. […]").

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 employ learning mechanisms for question-answer transformations (Agichtein et al., 2001; Radev et al., 2001), and also exploit large document collections such as the Web for finding answers (Brill et al., 2001; Kwok 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, trained on a corpus of 1 million question/answer pairs collected from the Web. Our evaluations show that our system achieves reasonable performance in terms of answer accuracy for a large variety of complex, non-factoid questions.

2 Beyond Factoid Question Answering

One of the first challenges to be faced in automatic question answering is the lexical and stylistic gap between the question string and the answer string. For factoid questions, these gaps are usually bridged by question reformulations, from simple rewrites (Brill et al., 2001), to more sophisticated paraphrases (Hermjakob et al., 2001), to question-to-answer translations (Radev et al., 2001). We ran several preliminary trials using various question reformulation techniques. We found out that in general, when complex questions are involved, reformulating the question (using either simple rewrites or question-answer term translations) more often hurts the performance than improves on it.

Another widely used technique in factoid QA is sentence parsing, along with question-type determination. As mentioned by Hovy et al. (2001), their hierarchical QA typology contains 79 nodes, which in many cases can be even further differentiated. While we acknowledge that QA typologies and hierarchical question types have the potential to be extremely useful beyond factoid QA, the volume of work involved is likely to exceed by orders of magnitude the one involved in the existing factoid QA typologies. We postpone such work for future endeavors.

The techniques we propose for handling our extended QA task are less linguistically motivated and more statistically driven. In order to have access to the right statistics, we first build a question-answer pair training corpus by mining FAQ pages from the Web, as described in Section 3. Instead of sentence parsing, we devise a statistical chunker that is used to transform a question into a phrase-based query (see Section 4). After a search engine uses the formulated query to return the *N* most relevant documents from the Web, an answer to the given question is found by computing an answer language model probability (indicating how similar the proposed answer is to answers seen in the training corpus), and an answer/question translation model probability (indicating how similar the proposed answer/question pair is to pairs seen in the training corpus). In Section 5 we describe the evaluations we performed in order to assess our system's performance, while in Section 6 we analyze some of the issues that negatively affected our system's performance.

3 A Question-Answer Corpus for FAQs

In order to employ the learning mechanisms 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., 1996) 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 an existing 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. Our URL collection contains 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 punctuation and lexical cues allowed us to retrieve most of the question/answer pairs, along with other noise data; a second precision-oriented pass used several filters, such as language identification, length constrains, and lexical cues to reduce the level of noise of the question/answer pair corpus. Using this method, we were able to collect a total of roughly 1 million question/answer pairs, exceeding by orders of magnitude the amount of data previously used for learning question/answer statistics.

4 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.

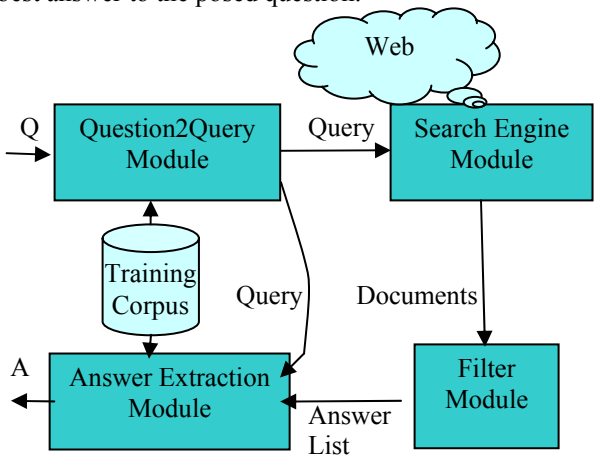

Figure 1: The QA system architecture

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 that present themselves as acceptable options to be evaluated.

4.1 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.

Our chunker uses the extracted collocation statistics to make an optimal chunking using a Dijkstra-style dynamic programming algorithm. In Figure 2 we present an example of the results returned by our statistical chunker. Important cues such as "differ from" and "herbal medications" are presented as phrases to the search engine, therefore increasing the recall of the search. Note that, unlike a segmentation offered by a parser (Hermjakob et al., 2001), 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"

Figure 2: Question segmentation into query using a statistical chunker

4.2 The SearchEngine Module

This module consists of a configurable interface with available off-the-shelf search engines. It currently supports MSNSearch and Google. Switching from one search engine to another allowed us to measure the impact of the IR engine on the QA task.

4.3 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 certain filtering steps. One 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. Other filtering steps performed by the Filter Module include 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 procedure described in Section 3), 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.

4.4 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. showed that techniques that did not bridge the lexical chasm were likely to perform worse than techniques that did.

For comparison purposes, we consider two different algorithms for our AnswerExtraction module: one that does not bridge the lexical chasm, based on N-gram co-occurrences between the question terms and the answer terms; and one that attempts to bridge 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.

4.4.1 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).

We implemented an answer extraction algorithm using the BLEU score of Papineni et al. (2002) as a means of assessing the overlap between the question and the proposed answers. For each potential answer, the overlap with the question was assessed with BLEU (with the brevity penalty set to penalize answers shorter than 3 times the length of the question). The best scoring potential answer was presented by the AnswerExtraction Module as the answer to the question.

4.4.2 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 3) 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 (2000) for the Information Retrieval task, as illustrated in Figure 3: 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.

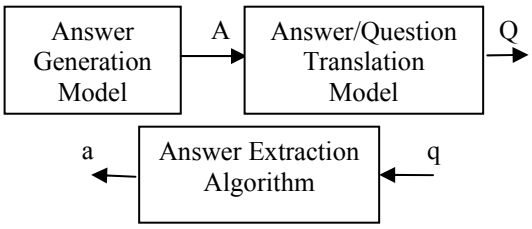

Figure 3: A noisy-channel model for answer extraction

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:

$$p(a \mid q,T) = \frac{p(q \mid a,T) \cdot p(a \mid T)}{p(q \mid T)} \qquad (1)$$

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:

$$a = \arg\max_{a} \underbrace{p(a \mid T)}_{question-independent} \cdot \underbrace{p(q \mid a,T)}_{question-dependent} \qquad (2)$$

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 3, 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 Black 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:

$$p(q \mid a) = \psi(m \mid n) \prod_{j=1}^{m} \left(\frac{n}{n+1} \left(\sum_{i=1}^{n} t(q_j \mid a_i) \cdot c(a_i \mid a) \right) + \frac{1}{n+1} t(q_j \mid NULL) \right) \quad (3)$$

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 (2000), an even simpler model than Model 1 can be devised by skewing the translation distribution $t(\cdot|\ a_i)$ such that all the probability mass goes to the term a_i. This simpler model is called Model 0. In Section 5 we evaluate the proficiency of both Model 1 and Model 0 in the answer extraction task.

5 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 asses 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*.

For the given set of Test questions, we estimated the performance of the system using the formula (|C|+.5|S|)/(|C|+|S|+|W|). This formula gives a score of 1 if the questions 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, in average, 1 out of 2 questions is answered correctly.

5.1 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 5.4).

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 4, 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.

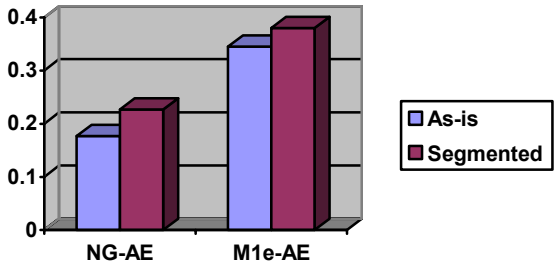

Figure 4: Evaluation of the Question2Query module

5.2 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 5, 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.

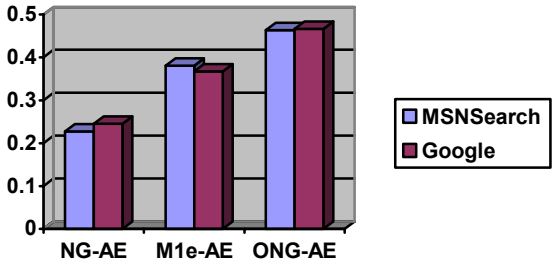

Figure 5: MSNSearch and Google give similar performance both in terms of realistic AE algorithms and oracle-based AE algorithms

5.3 Filter Module Evaluation

As mentioned in Section 4, 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 illustrated in Figure 6, 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, however, from considering the first 10 hits to considering the first 50 hits (0.46 to 0.49).

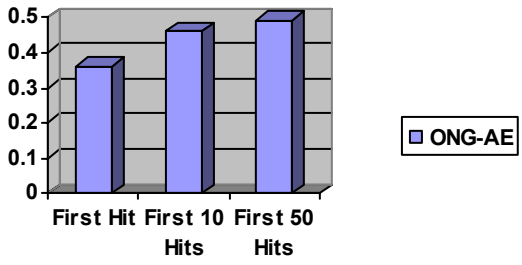

Figure 6: The scores obtained using the ONG-AE answer extraction algorithm for various N-best lists

5.4 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 7). 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 poorer 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 Model 1, 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 Model 0, 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 3 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 7 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).

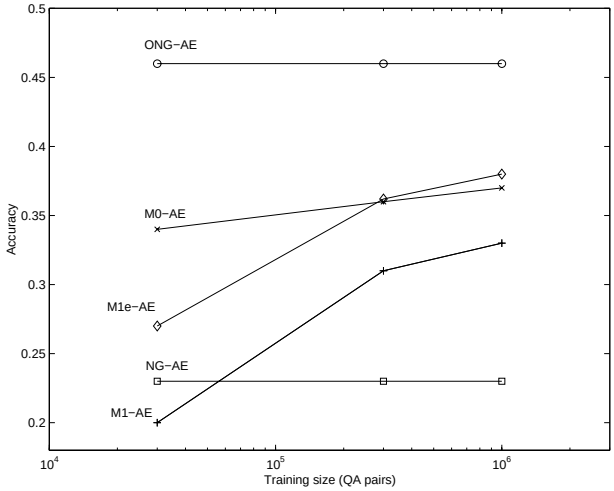

Figure 7: The performance of our QA system with various answer extraction algorithms and different training set sizes

6 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, such as:

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.

For those questions which were not answered correctly, we identified some of the most frequent causes which led to erroneous answers:

- answer was not in the retrieved pages (see the 46% performance ceiling given by the Oracle)
- 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. […]
- it 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.
- 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.

7 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., 2001; 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 Agichten, Steve Lawrence, and Luis Gravano. 2002. *Learning to Find Answers to Questions on the Web.* ACM Transactions on Internet Technology.

Adam L. Berger, John D. Lafferty. 1999. *Information Retrieval as Statistical Translation.* Proceedings of the SIGIR 1999, Berkeley, CA.

Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, Vibhu Mittal. 2000. *Bridging the Lexical Chasm: Statistical Approaches to Answer-Finding*. Research and Development in Information Retrieval, pages 192--199.

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, Andrew Ng. 2001. *Data-Intensive Question Answering*. *Proceedings of the TREC-2001Conference, NIST*. Gaithersburg, MD.

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.* Tech. Rep. TR-97-05, Dept. of Computer Science, University of Chicago.

Ted Dunning. 1993. *Accurate Methods for the Statistics of Surprise and Coincidence*. Computational Linguistics, Vol. 19, No. 1.

Abdessamad Echihabi and Daniel Marcu. 2003. *A Noisy-Channel Approach to Question Answering.* Proceedings of the ACL 2003. Sapporo, Japan.

Roxana Garju. 2003. *Automatic Detection of Causal Relations for Question Answering.* Proceedings of the ACL 2003, Workshop on "Multilingual Summarization and Question Answering - Machine Learning and Beyond", Sapporo, Japan.

Ulf Hermjakob, Abdessamad Echihabi, and Daniel Marcu. 2002. *Natural Language Based Reformulation Resource and Web Exploitation for Question Answering*. Proceedings of the TREC-2002 Conference, NIST. Gaithersburg, MD.

Abraham Ittycheriah and Salim Roukos. 2002. *IBM's Statistical Question Answering System-TREC 11*. Proceedings of the TREC-2002 Conference, NIST. Gaithersburg, MD.

Cody C. T. Kwok, Oren Etzioni, Daniel S. Weld. *Scaling Question Answering to the Web*. 2001. WWW10. Hong Kong.

Chin-Yew Lin and E.H. Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. Proceedings of the HLT/NAACL 2003. Edmonton, Canada.

Dan Moldovan, Sanda Harabagiu, Roxana Girju, Paul Morarescu, Finley Lacatusu, Adrian Novischi, Adriana Badulescu, Orest Bolohan. 2002. *LCC Tools for Question Answering.* Proceedings of the TREC-2002 Conference, NIST. Gaithersburg, MD.

Franz Joseph Och. 2003. *Minimum Error Rate Training in Statistical Machine Translation*. Proceedings of the ACL 2003. Sapporo, Japan.

Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu. 2002. *Bleu: a Method for Automatic Evaluation of Machine Translation*. Proceedings of the ACL 2002. Philadephia, PA.

Marius Pasca, Sanda Harabagiu, 2001. *The Informative Role of WordNet in Open-Domain Question Answering.* Proceedings of the NAACL 2001 Workshop on WordNet and Other Lexical Resources, Carnegie Mellon University. Pittsburgh, PA.

John M. Prager, Jennifer Chu-Carroll, Krysztof Czuba. 2001. *Use of WordNet Hypernyms for Answering What-Is Questions*. Proceedings of the TREC-2002 Conference, NIST. Gaithersburg, MD.

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*. Tenth International Conference onInformation and Knowledge Management. Atlanta, GA.

Jinxi Xu, Ana Licuanan, Jonathan May, Scott Miller, Ralph Weischedel. 2002. *TREC 2002 QA at BBN: Answer Selection and Confidence Estimation*. Proceedings of the TREC-2002 Conference, NIST. Gaithersburg, MD.