Proceedings of the 41st Annual Meeting of the Association
for Computational Linguistics, July 2003, pp. 16-23.

A Noisy-Channel Approach to Question Answering

Abdessamad Echihabi and Daniel Marcu
Information Sciences Institute
Department of Computer Science
University of Southern California
4676 Admiralty Way, Suite 1001
Marina Del Rey, CA 90292
{echihabi,marcu}@isi.edu

Abstract

We introduce a probabilistic noisy-channel model for question answering and we show how it can be exploited in the context of an end-to-end QA system. Our noisy-channel system outperforms a state-of-the-art rule-based QA system that uses similar resources. We also show that the model we propose is flexible enough to accommodate within one mathematical framework many QA-specific resources and techniques, which range from the exploitation of WordNet, structured, and semi-structured databases to reasoning, and paraphrasing.

1 Introduction

Current state-of-the-art Question Answering (QA) systems are extremely complex. They contain tens of modules that do everything from information retrieval, sentence parsing (Ittycheriah and Roukos, 2002; Hovy et al., 2001; Moldovan et al, 2002), question-type pinpointing (Ittycheriah and Roukos, 2002; Hovy et al., 2001; Moldovan et al, 2002), semantic analysis (Xu et al., Hovy et al., 2001; Moldovan et al, 2002), and reasoning (Moldovan et al, 2002). They access external resources such as the WordNet (Hovy et al., 2001, Pasca and Harabagiu, 2001, Prager et al., 2001), the web (Brill et al., 2001), structured, and semi-structured databases (Katz et al., 2001; Lin, 2002; Clarke, 2001). They contain feedback loops, ranking, and re-ranking modules. Given their complexity, it is often difficult (and sometimes impossible) to understand what contributes to the performance of a system and what doesn't.

In this paper, we propose a new approach to QA in which the contribution of various resources and components can be easily assessed. The fundamental insight of our approach, which departs significantly from the current architectures, is that, at its core, a QA system is a pipeline of only two modules:

- An IR engine that retrieves a set of M documents/N sentences that may contain answers to a given question Q.
- And an answer identifier module that given a question Q and a sentence S (from the set of sentences retrieved by the IR engine) identifies a sub-string S_A of S that is likely to be an answer to Q and assigns a score to it.

Once one has these two modules, one has a QA system because finding the answer to a question Q amounts to selecting the sub-string S_A of highest score. Although this view is not made explicit by QA researchers, it is implicitly present in all systems we are aware of.

In its simplest form, if one accepts a whole sentence as an answer ($S_A = S$), one can assess the likelihood that a sentence S contains the answer to a question Q by measuring the cosine similarity between Q and S. However, as research in QA demonstrates, word-overlap is not a good enough metric for determining whether a sentence contains the answer to a question. Consider, for example, the question "Who is the leader of France?" The sentence "Henri Hadjenberg, who is the leader of France's Jewish community, endorsed confronting the specter of the Vichy past" overlaps with all question terms, but it does not contain the correct answer; while the sentence "Bush later met with French President Jacques Chirac" does not overlap

with any question term, but it does contain the correct answer.

To circumvent this limitation of word-based similarity metrics, QA researchers have developed methods through which they first map questions and sentences that may contain answers in different spaces, and then compute the "similarity" between them there. For example, the systems developed at IBM and ISI map questions and answer sentences into parse trees and surface-based semantic labels and measure the similarity between questions and answer sentences in this syntactic/semantic space, using QA-motivated metrics. The systems developed by CYC and LCC map questions and answer sentences into logical forms and compute the "similarity" between them using inference rules. And systems such as those developed by IBM and BBN map questions and answers into feature sets and compute the similarity between them using maximum entropy models that are trained on question-answer corpora. From this perspective then, the fundamental problem of question answering is that of finding spaces where the distance between questions and sentences that contain correct answers is small and where the distance between questions and sentences that contain incorrect answers is large.

In this paper, we propose a new space and a new metric for computing this distance. Being inspired by the success of noisy-channel-based approaches in applications as diverse as speech recognition (Jelinek, 1997), part of speech tagging (Church, 1988), machine translation (Brown et al., 1993), information retrieval (Berger and Lafferty, 1999), and text summarization (Knight and Marcu, 2002), we develop a noisy channel model for QA. This model explains how a given sentence S_A that contains an answer sub-string A to a question Q can be rewritten into Q through a sequence of stochastic operations. Given a corpus of question-answer pairs (Q, S_A), we can train a probabilistic model for estimating the conditional probability $P(Q \mid S_A)$. Once the parameters of this model are learned, given a question Q and the set of sentences Σ returned by an IR engine, one can find the sentence $S_i \in \Sigma$ and an answer in it $A_{i,j}$ by searching for the $S_{i,A_{i,j}}$ that maximizes the conditional probability $P(Q \mid S_{i,A_{i,j}})$.

In Section 2, we first present the noisy-channel model that we propose for this task. In Section 3, we describe how we generate training examples. In Section 4, we describe how we use the learned models to answer factoid questions, we evaluate the performance of our system using a variety of experimental conditions, and we compare it with a rule-based system that we have previously used in several TREC evaluations. In Section 5, we demonstrate that the framework we propose is flexible enough to accommodate a wide range of resources and techniques that have been employed in state-of-the-art QA systems.

2 A noisy-channel for QA

Assume that we want to explain why "1977" in sentence S in Figure 1 is a good answer for the question "When did Elvis Presley die?" To do this, we build a noisy channel model that makes explicit how answer sentence parse trees are mapped into questions. Consider, for example, the automatically derived answer sentence parse tree in Figure 1, which associates to nodes both syntactic and shallow semantic, named-entity-specific tags. In order to rewrite this tree into a question, we assume the following generative story:

1. In general, answer sentences are much longer than typical factoid questions. To reduce the length gap between questions and answers and to increase the likelihood that our models can be adequately trained, we first make a "cut" in the answer parse tree and select a sequence of words, syntactic, and semantic tags. The "cut" is made so that every word in the answer sentence or one of its ancestors belongs to the "cut" and no two nodes on a path from a word to the root of the tree are in the "cut". Figure 1 depicts graphically such a cut.
2. Once the "cut" has been identified, we mark one of its elements as the answer string. In Figure 1, we decide to mark DATE as the answer string (A_DATE).
3. There is no guarantee that the number of words in the cut and the number of words in the question match. To account for this, we stochastically assign to every element s_i in a cut a fertility according to table $n(\phi \mid s_i)$. We delete elements of fertility 0 and duplicate elements of fertility 2, etc. With probability p_1 we also increment the fertility of an invisible

word NULL. NULL and fertile words, i.e. words with fertility strictly greater than 1 enable us to align long questions with short answers. Zero fertility words enable us to align short questions with long answers.

4. Next, we replace answer words (including the NULL word) with question words according to the table $t(q_i \mid s_j)$.
5. In the last step, we permute the question words according to a distortion table d, in order to obtain a well-formed, grammatical question.

The probability $P(Q \mid S_A)$ is computed by multiplying the probabilities in all the steps of our generative story (Figure 1 lists some of the factors specific to this computation.) The readers familiar with the statistical machine translation (SMT) literature should recognize that steps 3 to 5 are nothing but a one-to-one reproduction of the generative story proposed in the SMT context by Brown et al. (see Brown et al., 1993 for a detailed mathematical description of the model and the formula for computing the probability of an alignment and target string given a source string).[1]

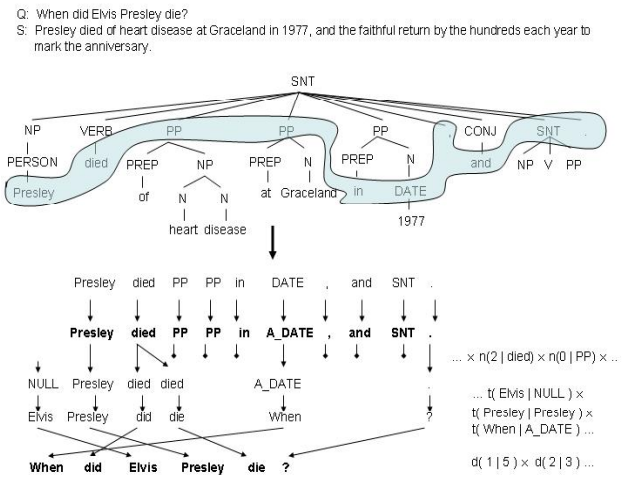

Figure 1: A generative model for Question answering

To simplify our work and to enable us exploit existing off-the-shelf software, in the experiments we carried out in conjunction with this paper, we assumed a flat distribution for the two steps in our generative story. That is, we assumed that it is equally likely to take any cut in the tree and equally likely to choose as Answer any syntactic/semantic element in an answer sentence.

3 Generating training and testing material

3.1 Generating training cases

Assume that the question-answer pair in Figure 1 appears in our training corpus. When this happens, we know that *1977* is the correct answer. To generate a training example from this pair, we tokenize the question, we parse the answer sentence, we identify the question terms and answer in the parse tree, and then we make a "cut" in the tree that satisfies the following conditions:

a) Terms overlapping with the question are preserved as surface text

b) The answer is reduced to its semantic or syntactic class prefixed with the symbol "A_"

c) Non-leaves, which don't have any question term or answer offspring, are reduced to their semantic or syntactic class.

d) All remaining nodes (leaves) are preserved as surface text.

Condition a) ensures that the question terms will be identified in the sentence. Condition b) helps learn answer types. Condition c) brings the sentence closer to the question by compacting portions that are syntactically far from question terms and answer. And finally the importance of lexical cues around question terms and answer motivates condition d). For the question-answer pair in Figure 1, the algorithm above generates the following training example:

Q: When did Elvis Presley die ?
S_A: *Presley died* PP PP *in* **A_DATE**, *and* SNT.

Figure 2 represents graphically the conditions that led to this training example being generated.

Our algorithm for generating training pairs implements deterministically the first two steps in our generative story. The algorithm is constructed so as to be consistent with our intuition that a generative process that makes the question and answer as similar-looking as possible is most likely to enable us learn a useful model. Each question-

[1] The distortion probabilities depicted in Figure 1 are a simplification of the distortions used in the IBM Model 4 model by Brown et al. (1993). We chose this watered down representation only for illustrative purposes. Our QA system implements the full-blown Model 4 statistical model described by Brown et al.

answer pair results in one training example. It is the examples generated through this procedure that we use to estimate the parameters of our model.

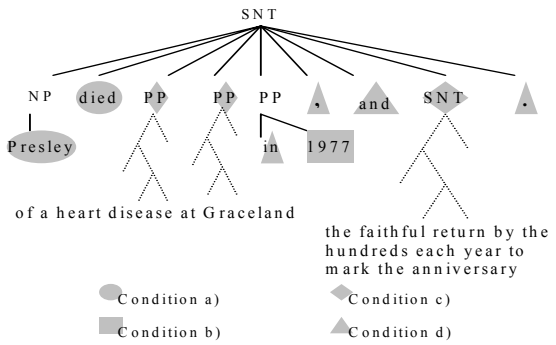

Figure 2: Generation of QA examples for training.

3.2 Generating test cases

Assume now that the sentence in Figure 1 is returned by an IR engine as a potential candidate for finding the answer to the question "When did Elvis Presley die?" In this case, we don't know what the answer is, so we assume that any semantic/syntactic node in the answer sentence can be the answer, with the exception of the nodes that subsume question terms and stop words. In this case, given a question and a potential answer sentence, we generate an exhaustive set of question-answer test cases, each test case labeling as answer (A_) a different syntactic/semantic node. Here are some of the test cases we consider for the question-answer pair in Figure 1:

```
   Q:  When did Elvis Presley die ?
..... SA1: Presley died A_PP PP PP , and SNT .
   Q:  When did Elvis Presley die ?
   SAi: Presley died PP PP in A_DATE, and
.....      SNT .
   Q:  When did Elvis Presley die ?
   SAj: Presley died PP PP PP , and NP
   return by A_NP NP .
```

If we learned a good model, we would expect it to assign a higher probability to $P(Q \mid S_{ai})$ than to $P(Q \mid S_{a1})$ and $P(Q \mid S_{aj})$.

4 Experiments

4.1 Training Data

For training, we use three different sets. (i) The TREC9-10 set consists of the questions used at TREC9 and 10. We automatically generate answer-tagged sentences using the TREC9 and 10 judgment sets, which are lists of answer-document pairs evaluated as either correct or wrong. For every question, we first identify in the judgment sets a list of documents containing the correct answer. For every document, we keep only the sentences that overlap with the question terms and contain the correct answer. (ii) In order to have more variation of sentences containing the answer, we have automatically extended the first data set using the Web. For every TREC9-10 question/answer pair, we used our Web-based IR to retrieve sentences that overlap with the question terms and contain the answer. We call this data set TREC9-10Web. (iii) The third data set consists of 2381 question/answer pairs collected from http://www.quiz-zone.co.uk. We use the same method to automatically enhance this set by retrieving from the web sentences containing answers to the questions. We call this data set Quiz-Zone. Table 1 shows the size of the three training corpora:

Training Set	# distinct questions	# question-answer pairs
TREC9-10	1091	18618
TREC9-10Web	1091	54295
Quiz-Zone	2381	17614

Table 1: Size of Training Corpora

To train our QA noisy-channel model, we apply the algorithm described in Section 3.1 to generate training cases for all QA pairs in the three corpora. To help our model learn that it is desirable to copy answer words into the question, we add to each corpus a list of identical dictionary word pairs w_i-w_i. For each corpus, we use GIZA (Al-Onaizan et al., 1999), a publicly available SMT package that implements the IBM models (Brown et al., 1993), to train a QA noisy-channel model that maps flattened answer parse trees, obtained using the "cut" procedure described in Section 3.1, into questions.

4.2 Test Data

We used two different data sets for the purpose of testing. The first set consists of the 500 questions used at TREC 2002; the second set consists of 500 questions that were randomly selected from the Knowledge Master (KM) repository (http://www.greatauk.com). The KM questions tend to be longer and quite different in style compared to the TREC questions.

4.3 A noisy-channel-based QA system

Our QA system is straightforward. It has only two modules: an IR module, and an answer-identifier/ranker module. The IR module is the same we used in previous participations at TREC. As the learner, the answer-identifier/ranker module is also publicly available – the GIZA package can be configured to automatically compute the probability of the Viterbi alignment between a flattened answer parse tree and a question.

For each test question, we automatically generate a web query and use the top 300 answer sentences returned by our IR engine to look for an answer. For each question Q and for each answer sentence S_i, we use the algorithm described in Section 3.2 to exhaustively generate all Q- $S_{i,A_{i,j}}$ pairs. Hence we examine all syntactic constituents in a sentence and use GIZA to assess their likelihood of being a correct answer. We select the answer $A_{i,j}$ that maximizes $P(Q \mid S_{i,A_{i,j}})$ for all answer sentences S_i and all answers $A_{i,j}$ that can be found in list retrieved by the IR module. Figure 3 depicts graphically our noisy-channel-based QA system.

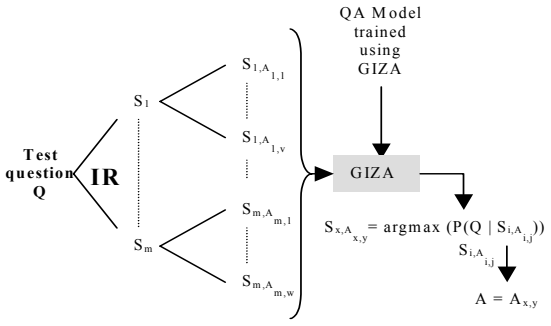

Figure 3: The noisy-channel-based QA system.

4.4 Experimental Results

We evaluate the results by generating automatically the mean reciprocal rank (MRR) using the TREC 2002 patterns and QuizZone original answers when testing on TREC 2002 and QuizZone test sets respectively. Our baseline is a state of the art QA system, QA-base, which was ranked from second to seventh in the last 3 years at TREC. To ensure a fair comparison, we use the same Web-based IR system in all experiments with no answer retrofitting. For the same reason, we use the QA-base system with the post-processing module disabled. (This module re-ranks the answers produced by QA-base on the basis of their redundancy, frequency on the web, etc.) Table 2 summarizes results of different combinations of training and test sets:

Trained on\Tested on	TREC 2002	KM
A = TREC9-10	0.325	0.108
B = A + TREC9-10Web	0.329	0.120
C = B + Quiz-Zone	0.354	0.132
QA-base	*0.291*	*0.128*

Table 2: Impact of training and test sets.

For the TREC 2002 corpus, the relatively low MRRs are due to the small answer coverage of the TREC 2002 patterns. For the KM corpus, the relatively low MRRs are explained by two factors: (i) for this corpus, each evaluation pattern consists of only one string – the original answer; (ii) the KM questions are more complex than TREC questions (*What piece of furniture is associated with Modred, Percival, Gawain, Arthur, and Lancelot?*).

It is interesting to see that using only the TREC9-10 data as training (system A in Table 2), we are able to beat the baseline when testing on TREC 2002 questions; however, this is not true when testing on KM questions. This can be explained by the fact that the TREC9-10 training set is similar to the TREC 2002 test set while it is significantly different from the KM test set. We also notice that expanding the training to TREC9-10Web (System B) and then to Quiz-Zone (System C) improved the performance on both test sets, which confirms that both the variability across answer tagged sentences (Trec9-10Web) and the abundance of distinct questions (Quiz-Zone) contribute to the diversity of a QA training corpus, and implicitly to the performance of our system.

5 Framework flexibility

Another characteristic of our framework is its flexibility. We can easily extend it to span other question-answering resources and techniques that have been employed in state-of-the art QA systems. In the rest of this section, we assess the impact of such resources and techniques in the context of three case studies.

5.1 Statistical-based "Reasoning"

The LCC TREC-2002 QA system (Moldovan et al., 2002) implements a reasoning mechanism for justifying answers. In the LCC framework,

questions and answers are first mapped into logical forms. A resolution-based module then proves that the question logically follows from the answer using a set of axioms that are automatically extracted from the WordNet glosses. For example, to prove the logical form of "What is the age of our solar system?" from the logical form of the answer "The solar system is 4.6 billion years old.", the LCC theorem prover shows that the atomic formula that corresponds to the question term "age" can be inferred from the atomic formula that corresponds to the answer term "old" using an axiom that connects "old" and "age", because the WordNet gloss for "old" contains the word "age". Similarly, the LCC system can prove that "Voting is mandatory for all Argentines aged over 18" provides a good justification for the question "What is the legal age to vote in Argentina?" because it can establish through logical deduction using axioms induced from WordNet glosses that "legal" is related to "rule", which in turn is related to "mandatory"; that "age" is related to "aged"; and that "Argentine" is related to "Argentina". It is not difficult to see by now that these logical relations can be represented graphically as alignments between question and answer terms (see Figure 4).

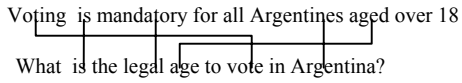

Figure 4: Gloss-based reasoning as word-level alignment.

The exploitation of WordNet synonyms, which is part of many QA systems (Hovy et al., 2001; Prager et al., 2001; Pasca and Harabagiu, 2001), is a particular case of building such alignments between question and answer terms. For example, using WordNet synonymy relations, it is possible to establish a connection between "U.S." and "United States" and between "buy" and "purchase" in the question-answer pair (Figure 5), thus increasing the confidence that the sentence contains a correct answer.

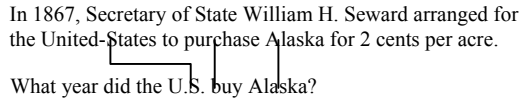

Figure 5: Synonym-based alignment.

The noisy channel framework we proposed in this paper can approximate the reasoning mechanism employed by LCC and accommodate the exploitation of gloss- and synonymy-based relations found in WordNet. In fact, if we had a very large training corpus, we would expect such connections to be learned automatically from the data. However, since we have a relatively small training corpus available, we rewrite the WordNet glosses into a dictionary by creating word-pair entries that establish connections between all Wordnet words and the content words in their glosses. For example, from the word "age" and its gloss "a historic period", we create the dictionary entries "age - historic" and "age – period". To exploit synonymy relations, for every WordNet synset S_i, we add to our training data all possible combinations of synonym pairs $W_{i,x}$-$W_{i,y}$.

Our dictionary creation procedure is a crude version of the axiom extraction algorithm described by Moldovan et al. (2002); and our exploitation of the glosses in the noisy-channel framework amounts to a simplified, statistical version of the semantic proofs implemented by LCC. Table 3 shows the impact of WordNet synonyms (WNsyn) and WordNet glosses (WNgloss) on our system. Adding WordNet synonyms and glosses improved slightly the performance on the KM questions. On the other hand, it is surprising to see that the performance has dropped when testing on TREC 2002 questions.

Trained on\Tested on	TREC 2002	KM
C	0.354	0.132
C+WNsyn	0.345	0.138
C + WNgloss	0.343	0.136

Table 3: WordNet synonyms and glosses impact.

5.2 Question reformulation

Hermjakob et al. (2002) showed that reformulations (syntactic and semantic) improve the answer pinpointing process in a QA system. To make use of this technique, we extend our training data set by expanding every question-answer pair Q-S_A to a list (Q_r-S_A), $Q_r \subset \Theta$ where Θ is the set of question reformulations.[2] We also expand in a similar way the answer candidates in the test corpus. Using reformulations improved the

[2] We are grateful to Ulf Hermjakob for sharing his reformulations with us.

performance of our system on the TREC 2002 test set while it was not beneficial for the KM test set (see Table 4). We believe this is explained by the fact that the reformulation engine was fine tuned on TREC-specific questions, which are significantly different from KM questions.

Trained on\Tested on	TREC 2002	KM
C	0.354	0.132
C+reformulations	0.365	0.128

Table 4: Reformulations impact.

5.3 Exploiting data in structured -and semi-structured databases

Structured and semi-structured databases were proved to be very useful for question-answering systems. Lin (2002) showed through his federated approach that 47% of TREC-2001 questions could be answered using Web-based knowledge sources. Clarke et al. (2001) obtained a 30% improvement by using an auxiliary database created from web documents as an additional resource. We adopted a different approach to exploit external knowledge bases.

In our work, we first generated a natural language collection of factoids by mining different structured and semi-structured databases (World Fact Book, Biography.com, WordNet…). The generation is based on manually written question-factoid template pairs, which are applied on the different sources to yield simple natural language question-factoid pairs. Consider, for example, the following two factoid-question template pairs:

Q_{t1}: What is the capital of _c?
S_{t1}: The capital of _c is ***capital(_c)***.
Q_{t2}: How did _p die?
S_{t2}: _p died of **causeDeath(_p)**.

Using extraction patterns (Muslea, 1999), we apply these two templates on the World Fact Book database and on biography.com pages to instantiate question and answer-tagged sentence pairs such as:

Q_1: What is the capital of Greece?
S_1: The capital of Greece is **Athens.**
Q_2: How did Jean-Paul Sartre die?
S_2: Jean-Paul Sartre died of **a lung ailment.**

These question-factoid pairs are useful both in training and testing. In training, we simply add all these pairs to the training data set. In testing, for every question Q, we select factoids that overlap sufficiently enough with Q as sentences that potentially contain the answer. For example, given the question "Where was Sartre born?" we will select the following factoids:

1-Jean-Paul Sartre was born in 1905.
2-Jean-Paul Sartre died in 1980.
3-Jean-Paul Sartre was born in Paris.
4-Jean-Paul Sartre died of a lung ailment.

Up to now, we have collected about 100,000 question-factoid pairs. We found out that these pairs cover only 24 of the 500 TREC 2002 questions. And so, in order to evaluate the value of these factoids, we reran our system C on these 24 questions and then, we used the question-factoid pairs as the only resource for both training and testing as described earlier (System D). Table 5 shows the MRRs for systems C and D on the 24 questions covered by the factoids.

System	24 TREC 2002 questions
C	0.472
D	0.812

Table 5: Factoid impact on system performance.

It is very interesting to see that system D outperforms significantly system C. This shows that, in our framework, in order to benefit from external databases, we do not need any additional machinery (question classifiers, answer type identifiers, wrapper selectors, SQL query generators, etc.) All we need is a one-time conversion of external structured resources to simple natural language factoids. The results in Table 5 also suggest that collecting natural language factoids is a useful research direction: if we collect all the factoids in the world, we could probably achieve much higher MRR scores on the entire TREC collection.

6 Conclusion

In this paper, we proposed a noisy-channel model for QA that can accommodate within a unified framework the exploitation of a large number of resources and QA-specific techniques. We believe that our work will lead to a better understanding of the similarities and differences between the approaches that make up today's QA research landscape. We also hope that our paper will reduce the high barrier to entry that is explained by the complexity of current QA systems and increase the number of researchers working in this field: because our QA system uses only publicly available software components (an IR engine; a

parser; and a statistical MT system), it can be easily reproduced by other researchers.

However, one has to recognize that the reliance of our system on publicly available components is not ideal. The generative story that our noisy-channel employs is rudimentary; we have chosen it only because we wanted to exploit to the best extent possible existing software components (GIZA). The empirical results we obtained are extremely encouraging: our noisy-channel system is already outperforming a state-of-the-art rule-based system that took many person years to develop. It is remarkable that a statistical machine translation system can do so well in a totally different context, in question answering. However, building dedicated systems that employ more sophisticated, QA-motivated generative stories is likely to yield significant improvements.

Acknowledgments. This work was supported by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program under contract number MDA908-02-C-0007.

References

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. 1999. Statistical machine translation. *Final Report, JHU Summer Workshop.*

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

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

Kenneth W. Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. *In Proceedings of the Second Conference on Applied Natural Language Processing, Austin, TX.*

Charles L. A. Clarke, Gordon V. Cormack, Thomas R. Lynam, C. M. Li, G. L. McLearn. 2001. Web Reinforced Question Answering (MultiText Experiments for TREC 2001). In *Proceedings of the TREC-2001Conference, NIST. Gaithersburg, MD.*

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

Edward H. Hovy, Ulf Hermjakob, Chin-Yew Lin. 2001. The Use of External Knowledge in Factoid QA. In *Proceedings of the TREC-2001 Conference, NIST. Gaithersburg, MD.*

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

Frederick Jelinek. 1997. Statistical Methods for Speech Recognition. MIT Press, Cambridge, MA.

Boris Katz, Deniz Yuret, Sue Felshin. 2001. Omnibase: A universal data source interface. In *MIT Artificial Intelligence Abstracts.*

Kevin Knight, Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence 139(1): 91-107.*

Jimmy Lin. 2002. The Web as a Resource for Question Answering: Perspective and Challenges. In *LREC 2002, Las Palmas, Canary Islands, Spain.*

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 TREC-2002 Conference, NIST. Gaithersburg, MD.*

Ion Muslea. 1999. Extraction Patterns for Information Extraction Tasks: A Survey. In *Proceedings of Workshop on Machine Learning and Information Extraction (AAAI-99), Orlando, FL.*

Marius Pasca, Sanda Harabagiu, 2001. The Informative Role of WordNet in Open-Domain Question Answering. In *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. In *Proceedings of the TREC-2002 Conference, NIST. Gaithersburg, MD.*

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