Statistical QA - Classifier vs. Re-ranker: What's the difference?

Deepak Ravichandran, Eduard Hovy, and Franz Josef Och
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292
{ravichan,hovy,och}@isi.edu

Abstract

In this paper, we show that we can obtain a good baseline performance for Question Answering (QA) by using only 4 simple features. Using these features, we contrast two approaches used for a Maximum Entropy based QA system. We view the QA problem as a classification problem and as a re-ranking problem. Our results indicate that the QA system viewed as a re-ranker clearly outperforms the QA system used as a classifier. Both systems are trained using the same data.

1 Introduction

Open-Domain factoid Question Answering (QA) is defined as the task of answering fact-based questions phrased in Natural Language. Examples of some question and answers that fall in the fact-based category are:

1. What is the capital of Japan? - Tokyo
2. What is acetaminophen? - Non-aspirin pain killer
3. Where is the Eiffel Tower? - Paris

The architecture of most of QA systems consists of two basic modules: the information retrieval (IR) module and the answer pinpointing module. These two modules are used in a typical pipeline architecture.

For a given question, the IR module finds a set of relevant segments. Each segment typically consists of at most R sentences[1]. The answer pinpointing module processes each of these segments and finds the appropriate answer phrase. Evaluation of a QA system is judged on the basis on the final output answer and the corresponding evidence provided by the segment.

This paper focuses on the answer pinpointing module. Typical QA systems perform re-ranking of candidate answers as an important step in pinpointing. The goal is to rank the most likely answer first by using either symbolic or statistical methods. Some QA systems make use of statistical answer pinpointing (Xu et. al, 2002; Ittycheriah, 2001; Ittycheriah and Salim, 2002) by treating it as a classification problem. In this paper, we cast the pinpointing problem in a statistical framework and compare two approaches, classification and re-ranking.

2 Statistical Answer Pinpointing

2.1 Answer Modeling

The answer-pinpointing module gets as input a question q and a set of possible answer candidates $\{a_1 a_2 ... a_A\}$. It outputs one of the answer $a \in \{a_1 a_2 ... a_A\}$ from the candidate answer set. We consider two ways of modeling this problem.

One approach is the traditional classification view (Ittycheriah, 2001) where we present each Question-Answer pair to the classifier which classifies it as either correct answer (true) or incorrect answer (false), based on some evidence (features).

In this case, we model $P(c \mid a, \{a_1 a_2 .. a_A\}, q)$. Here, $c = \{\text{true, false}\}$ signifies the correctness of the answer a with respect to the question q.

The probability $P(c \mid a, \{a_1 a_2 .. a_A\}, q)$ for each QA pair is modeled independently of other such pairs. Thus, for the same question, many QA pairs are presented to the classifier as independ-

[1] In our experiments we use R=1

ent events (histories). If the training corpus contains Q questions with A answers for each question, the total number of events (histories) would be equal to Q·A with two classes (futures) (correct or incorrect answer) for each event. Once the probabilities $P(c|a,\{a_1 a_2..a_A\},q)$ have been computed, the system has to return the best answer. The following decision rule is used:

$$\hat{a} = \arg\max_a [P(true \mid a, \{a_1 a_2 ... a_A\}, q)]$$

Another way of viewing the problem is as a re-ranking task. This is possible because the QA task requires the identification of only one correct answer, instead of identifying all the correct answer in the collection. In this case, we model $P(a \mid \{a_1 a_2 ... a_A\}, q)$. If the training corpus contains Q questions with A answers for each question, the total number of events (histories) would be equal to Q, with A classes (futures). This view requires the following decision-rule to identify the answer that seems most promising:

$$\hat{a} = \arg\max_a [P(a \mid \{a_1 a_2 ... a_A\}, q)]$$

In summary,

	Classifier	Re-ranker
#Events (Histories)	Q·A	Q
#Classes (Futures) per event	2	A

where,
Q = total number of questions.
A = total number of answer chunks considered for each question.

2.2 Maximum Entropy formulation

We use Maximum Entropy to model the given problem both as a classifier and a re-ranker. We define M feature functions, $f_m(a,\{a_1 a_2 ... a_A\},q), m = 1,....,M$, that may be useful in characterizing the task. Della Pietra et. al (1995) contains good description of Maximum Entropy models.

We model the classifier as follows:

$$P(c \mid a,\{a_1 a_2 .. a_A\},q) = \frac{\exp[\sum_{m=1}^{M} \lambda_{m,c} f_m(a,\{a_1 a_2 .. a_A\},q)]}{\sum_{c'} \exp[\sum_{m=1}^{M} \lambda_{m,c'} f_m(a,\{a_1 a_2 .. a_A\},q)]}$$

where,
$\lambda_{m,c}; m = 1,...., M; c = \{true, false\}$ are the model parameters.

The decision rule for choosing the best answer is:

$$\hat{a} = \arg\max_a [P(true \mid a, \{a_1 a_2 ... a_A\}, q)]$$

$$= \arg\max_a [\sum_{m=1}^{M} \lambda_{m,true} f_m(a, \{a_1 a_2 ... a_A\}, q)]$$

The above decision rule requires comparison of different probabilities of the form $P(true \mid a, \{a_1 a_2 ... a_A\}, q)$. However, these probabilities are modeled as independent events (histories) in the classifier and hence the training criterion does not make them directly comparable.

For the re-ranker, we model the probability as:

$$P(a \mid \{a_1 a_2 ... a_A\}, q) = \frac{\exp[\sum_{m=1}^{M} \lambda_m f_m(a,\{a_1 a_2 ... a_A\},q)]}{\sum_{a'} \exp[\sum_{m=1}^{M} \lambda_m f_m(a',\{a_1 a_2 ... a_A\},q)]}$$

where,
$\lambda_m; m = 1,...., M$ are the model parameters.
Note that for the classifier the model parameters are $\lambda_{m,c}$, whereas for the re-ranker they are λ_m. This is because for the classifier, each feature function has different weights associated with each class (future). Hence, the classifier has twice the model parameters as compared to the re-ranker.
The decision rule for the re-ranker is given by:

$$\hat{a} = \arg\max_a [P(a \mid \{a_1 a_2 ... a_A\}, q)]$$

.

$$= \arg\max_a [\sum_{m=1}^{M} \lambda_m f_m(a, \{a_1 a_2 ... a_A\}, q)]$$

The re-ranker makes the probabilities $P(a \mid \{a_1 a_2 ... a_A\}, q)$, considered for the decision rule, directly comparable against each other, by incorporating them into the training criterion itself. Table 1 summarizes the differences of the two models.

Table 1 : Model comparison between a Classifier and Re-ranker

	Classifier	Re-Ranker
Modeling Equation	$P(c \mid a,\{a_1 a_2..a_A\},q)=\frac{\exp[\sum_{m=1}^{M}\lambda_{m,c} f_m(a,\{a_1 a_2..a_A\},q)]}{\sum_{c'}\exp[\sum_{m=1}^{M}\lambda_{m,c'} f_m(a,\{a_1 a_2..a_A\},q)]}$	$P(a \mid \{a_1 a_2...a_A\},q)=\frac{\exp[\sum_{m=1}^{M}\lambda_m f_m(a,\{a_1 a_2..a_A\},q)]}{\sum_{a'}\exp[\sum_{m=1}^{M}\lambda_m f_m(a',\{a_1 a_2..a_A\},q)]}$
Decision Rule	$\hat{a}=\arg\max_a\{P(true \mid a,\{a_1 a_2...a_A\},q)\}$ $=\arg\max_a[\sum_{m=1}^{M}\lambda_m f_{m,true}(a,\{a_1 a_2...a_A\},q)]$	$\hat{a}=\arg\max_a[P(a \mid \{a_1 a_2...a_A\},q)]$. $=\arg\max_a[\sum_{m=1}^{M}\lambda_m f_m(a,\{a_1 a_2...a_A\},q)]$

2.3 Feature Functions

Using above formulation to model the probability distribution we need to come up with features f_j. We use only four basic feature functions for our system.

1. Frequency: It has been observed that the correct answer has a higher frequency (Magnini et al.; 2002) in the collection of answer chunks (C). Hence we count the number of time a potential answer occurs in the IR output and use its logarithm as a feature. This is a positive continuous valued feature.
2. Expected Answer Class: Most of the current QA systems employ some type of Answer Class Identification module. Thus questions like "When did Bill Clinton go to college?", would be identified as a question asking about a time (or a time period), "Where is the sea of tranquility?" would be identified as a question asking for a location. If the answer class matches the expected answer class (derived from the question by the answer identification module) this feature fires (i.e., it has a value of 1). Details of this module are explained in Hovy et al. (2002). This is a binary-valued feature.
3. Question Word Absent: Usually a correct answer sentence contains a few of the question words. This feature fires if the candidate answer does not contain any of the question words. This is also a binary valued feature.
4. Word Match: It is the sum of ITF[2] values for the words in the question that matches identically with the words in the answer sentence. This is a positive continuous valued feature.

2.4 Training

We train our Maximum Entropy model using Generalized Iterative scaling (Darroch and Ratcliff, 1972) approach by using YASMET[3] .

3 Evaluation Metric

The performance of the QA system is highly dependent on the performance of the two individual modules IR and answer-pinpointing. The system would have excellent performance if both have good accuracy. Hence, we need a good evaluation metric to evaluate each of these components individually. One standard metric for IR is recall and precision. We can modify this metric for QA as follows:

[2] ITF = Inverse Term Frequency. We take a large independent corpus & estimate ITF(W) =1/(count(W)), where W = Word.

[3] YASMET. (Yet Another Small Maximum Entropy Toolkit) http://www-i6.informatik.rwth-aachen.de/Colleagues/och/software/YASMET.html

```
Question:
1395 Who is Tom Cruise married to ?

IR Output:
1 Tom Cruise is married to actress Nicole Kidman and they have two adopted children .
2 Tom Cruise is married to Nicole Kidman .
.
.

Output of Chunker: (The number to the left of each chunk records the IR sentence from
which that particular chunk came)
1  Tom Cruise
1  Tom
1  Cruise
1  is married
1  married
1  actress Nicole Kidman and they
1  actress Nicole Kidman
1  actress
1  Nicole Kidman
1  Nicole
1  Kidman
1  they
1  two adopted children
1  two
1  adopted
1  children
2  Tom Cruise
2  Tom
2  Cruise
2  is married
2  married
2  Nicole Kidman
2  Nicole
2  Kidman
.
.
```

Figure 1 : Candidate answer extraction for a question.

$$\text{Recall} = \frac{\#\text{ relevant answer segment returned}}{\#\text{ Total relevant answer segments}}$$

$$\text{Precision} = \frac{\#\text{ relevant answer segments returned}}{\#\text{ Total segments returned}}$$

It is almost impossible to measure recall because the IR collection is typically large and involves several hundreds of thousands of documents. Hence, we evaluate our IR by only the precision measure at top N segments. This method is actually a rather sloppy approximation to the original recall and precision measure. Questions with fewer correct answers in the collection would have a lower precision score as compared to questions with many answers. Similarly, it is unclear how one would evaluate answer questions with No Answer (NIL) in the collection using this metric. All these questions would have zero precision from the IR collection.

The answer-pinpointing module is evaluated by checking if the answer returned by the system as the top ranked (#1) answer is correct/incorrect with respect to the IR collection and the true answer. Hence, if the IR system fails to return even a single sentence that contains the correct answer for the given question, we do not penalize the answer-pinpointing module. It is again unclear how to evaluate questions with No answer (NIL). (Here, for our experiments we attribute the error to the IR module.)

Finally, the combined system is evaluated by using the standard technique, wherein the Answer (ranked #1) returned by the system is judged to be either correct or incorrect and then the average is taken.

4 Experiments

4.1 Framework

Information Retrieval

For our experiments, we use the Web search engine AltaVista. For every question, we remove stop-words and present all other question words as query to the Web search engine. The top relevant documents are downloaded. We apply a sentence segmentor, and identify those sentences that have high ITF overlapping words with the given question. The sentences are then re-ranked accordingly and only the top K sentences (segments) are presented as output of the IR system.

Candidate Answer Extraction

For a given question, the IR returns top K segments. For our experiments a segment consists of one sentence. We parse each of the sentences and obtain a set of chunks, where each chunk is a node of the parse tree. Each chunk is viewed as a potential answer. For our experiments we restrict the number of potential answers to be at most 5000. We illustrate this process in Figure 1.

Training/Test Data

Table 2 : Training size and sources.

	Training + Validation	Test
Question collection	TREC 9 + TREC 10	TREC11
Total questions	1192	500

We use the TREC 9 and TREC 10 data sets for training and the TREC 11 data set for testing. We initially apply the IR step as described above and obtain a set of at most 5000 answers. For each such answer we use the pattern file supplied by NIST to tag answer chunks as either correct (1) or incorrect (0). This is a very noisy way of tagging data. In some cases, even though the answer chunk may be tagged as correct it may not be supported by the accompanying sentence, while in other cases, a correct chunk may be graded as incorrect, since the pattern file list did not represent a exhaustive list of answers. We set aside 20% of the training data for validation.

4.2 Classifier vs. Re-Ranker

We evaluate the performance of the QA system viewed as a classifier (with a post-processing step) and as a re-ranker. In order to do a fair evaluation of the system we test the performance of the QA system under varying conditions of the output of the IR system. The results are shown in Table 3.

The results should be read in the following way: We use the same IR system. However, during each run of the experiment we consider only the top K sentences returned by the IR system K={1,10,50,100,150,200}. The column "correct" represents the number of questions the entire QA (IR + re-ranker) system answered correctly. "IR Loss" represents the average number of questions for which the IR failed completely (i.e., the IR did not return even a single sentence that contains the correct answer). The IR precision is the precision of the IR system for the number of sentences considered. Answer-pinpointing performance is based on the metric described above. Finally, the overall score is the score of the entire QA system. (i.e., precision at rank#1).

The "Overall Precision" column indicates that the re-ranker clearly outperforms the classifier. However, it is also very interesting to compare the performance of the re-ranker "Overall Precision" with the "Answer-Pinpointing precision". For example, in the last row, for the re-ranker the "Answer-Pinpointing Precision" is 0.5182 whereas the "Overall Precision" is only 0.34. The difference is due to the performance of the poor performance of the IR system ("IR Loss" = 0.344).

Table 3 : Results for Classifier and Re-ranker under varying conditions of IR.

IR Sentences	Total questions	IR Precision	IR Loss	Answer-Pinpointing Precision		Number Correct		Overall Precision	
				Classifier	Re-ranker	Classifier	Re-ranker	Classifier	Re-ranker
1	500	0.266	0.742	0.0027	0.3565	29	46	0.058	0.092
10	500	0.2018	0.48	0.0016	0.4269	7	111	0.014	0.222
50	500	0.1155	0.386	0.0015	0.4885	6	150	0.012	0.3
100	500	0.0878	0.362	0.0015	0.5015	5	160	0.01	0.32
150	500	0.0763	0.35	0.0015	0.5138	5	167	0.01	0.334
200	500	0.0703	0.344	0.0015	0.5182	3	170	0.01	0.34

IR Sentences = Total IR sentences considered for every question
IR Precision = Precision @ (IR Sentences)
IR Loss = (Number of Questions for which the IR did not produce a single answer)/(Total Questions)
Overall Precision = (Number Correct)/(Total Questions)

4.3 Oracle IR system

In order to determine the performance of the answer pinpointing module alone, we perform the so-called oracle IR experiment. Here, we present to the answer pinpointing module only those sentences from IR that contain an answer[4]. The task of the answer pinpointing module is to pick out of the correct answer from the given collection. We report results in Table 4. In these results too the re-ranker has better performance as compared to the classifier. However, as we see from the results, there is a lot of room for improvement for the re-ranker system, even with a perfect IR system.

5 Discussion

Our experiments clearly indicate that the QA system viewed as a re-ranker outperforms the QA system viewed as a classifier. The difference stem from the following reasons:

1. The classification training criteria work on a more difficult objective function of trying to find whether each candidate answer answers the given question, as opposed to trying to find the best answer for the given question. Hence, the same feature set that works for the re-ranker need not work for the classifier. The feature set used in this problem is not good enough to help the classifier distinguish between correct and incorrect answers for the given question (even though it is good for the re-ranker to come up with the best answer).
2. The comparison of probabilities across different events (histories) for the classifier, during the decision rule process, is problematic. This is because the probabilities, which we obtain after the classification approach, are only a poor estimate of the true probability. The re-ranker, however, directly allows these probabilities to be comparable by incorporating them into the model itself.
3. The QA system viewed as a classifier suffers from the problem of a highly unbalanced data set. We have less than 1% positive examples and more than 99% negative examples (we had almost 4 million training data events) in the problem. Ittycheriah (2001), and Ittycheriah and Roukos (2002), use a more controlled environment for training their system. They have 23% positive examples and 77% negative examples. They prune out most of the incorrect answer initially, using a pre-processing step by using either a rule-based system (Ittycheriah, 2001) or a statistical system (Ittycheriah et al., 2002); and hence obtain a much more manageable distribution in the training phase of the Maximum Entropy model.

[4] This was performed by extracting all the sentences that were judged to have the correct answer by human evaluators during the TREC 2002 evaluations.

Table 4 : Performance with a perfect IR system

Total questions	IR precision	Answer-Pinpointing Precision	
		Classifier	Re-ranker
429	1.0	0.156	0.578

6 Conclusion

The re-ranker system is very robust in handling large amounts of data and still produces reasonable results. There is no need for a major preprocessing step (for eliminating undesirable incorrect answers from the training) or the postprocessing step (for selecting the most promising answer.)

We also consider it significant that a QA system with just 4 features (viz. Frequency, Expected Answer Type, Question word absent, and ITF word match) is a good baseline system and performs better than the median performance of all the QA systems in the TREC 2002 evaluations[5].

Ittycheriah (2001), and Ittycheriah and Roukos (2002) have shown good results by using a range of features for Maximum Entropy QA systems. Also, the results indicate that there is scope for research in IR for QA systems. The QA system has an upper ceiling on performance due to the quality of the IR system. The QA community has yet to address these problems in a principled way, and the IR details of most of the system are hidden behind the complicated system architecture.

The re-ranking model basically changes the objective function for training and the system is directly optimized on the evaluation function criteria (though still using Maximum Likelihood training). Also this approach seems to be very robust to noisy training data and is highly scalable.

Acknowledgements.

This work was supported by the Advance Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program under contract number MDA908-02-C-007. The authors wish to express particular gratitude to Dr. Abraham Ittycheriah, both for his supervision and education of the first author during his summer visit to IBM TJ Watson Research Center in 2002 and for his thoughtful comments on this paper, which was inspired by his work.

[5] However, since the IR system used here was from the Web, our results are not directly comparable with the TREC systems.

References

Darroch, J. N., and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43:1470–1480.

Hermjakob, U. 1997. Learning Parse and Translation Decisions from Examples with Rich Context. *Ph.D. Dissertation*, University of Texas at Austin, Austin, TX.

Hovy, E.H., U. Hermjakob, D. Ravichandran. 2002. A Question/Answer Typology with Surface Text Patterns. *Proceedings of the DARPA Human Language Technology Conferenc*,. San Diego, CA, 247–250.

Ittycheriah, A. 2001. Trainable Question Answering System. *Ph.D. Dissertation*, Rutgers, The State University of New Jersey, New Brunswick, NJ.

Ittycheriah., A., and S. Roukos. 2002. IBM'S Question Answering System-TREC-11. *Proceedings of TREC 2002*, NIST, MD, 394–401.

Magnini, B, M. Negri, R. Prevete, and H. Tanev. 2002. Is it the Right Answer? Exploiting Web Redundancy for Answer Validation. Proceedings of the 40th Meeting of the *Association of Computational Linguistics*, Philadelphia, PA, 425–432.

Della Pietra, S., V. Della Pietra, and J. Lafferty. 1995. Inducing Features of Random Fields, *Technical Report* Department of Computer Science, Carnegie-Mellon University, CMU–CS-95–144.

Xu, J., A. J. Licuanan, S. May, R. Miller, and R. Weischedel. 2002. TREC2002QA at BBN: Answer Selection and Confidence Estimation. *Proceedings of TREC 2002.* NIST MD. 290–295