The University of Michigan at TREC2002: Question Answering and Novelty tracks

Hong Qi, Jahna Otterbacher, Adam Winkel, Dragomir R. Radev
University of Michigan
Ann Arbor, MI 48109
{hqi,jahna,winkela,radev}@umich.edu

November 3, 2002

1 Introduction

The University of Michigan participated in two evaluations this year. In the Question Answering Track, we entered three different versions of our system, NSIR, previously described in [1]. For the Novelty Track, we modified our multi-document summarizer, MEAD (`www.summarization.com/mead`) and submitted five runs with different input parameters.

2 Question Answering Track

We submitted three runs to the main task of the TREC Question Answering track this year. The system that we describe in this report is NSIR, a QA system being developed at the University of Michigan. NSIR uses a fine question taxonomy (2.1), extracts candidate answers along with nine features (2.2), and ranks the answers (2.3). We will show the architecture of the NSIR system (2.4). The official results from TREC will be provided, and we will also discuss how we plan to improve the performance of NSIR system (2.5).

2.1 Question Classification

To identify the semantic type of a question is a critical step when a QA system attempts to return phrasal answers. Our working assumption is that a finer taxonomy can better support answer extraction. We developed a hierarchical taxonomy that includes 141 different types. We believe that different types of questions should have different treatments. Therefore, we try to construct a tuned method to answer questions of each different type in the finer taxonomy.

By categorizing question "Where is the capital of Spain?" into *Captial* type instead of a more general type *Place*, for example, we could then use a pre-defined list of capital cities of each country to help answer this question. For another example, the target answer of a *Mountain* type question is likely to have the word "Mountain" or "Mt." followed by a mountain name which usually would be tagged as NNPs, i.e., proper nouns, by a typical part-of-speech tagger.

However, one drawback of using a finer taxonomy is that sometimes it is hard to classify a question into one single class. For example, the target answer of "Who won the Superbowl in 1982?" can be "San Francisco" which is a location or "49ers" which is the name of the team. Classifying each question into one single type may potentially lead to false negative cases, or, false positive answers, which is even worse. We adopted a probabilistic question classifier, which assigns normalized weights for each possible question type if the target answer could be in more than one categories. The probabilities of being each question type will be taken into account when ranking all the potential answers. Therefore, in our example, both "San Francisco" and "49ers" will be boosted if the question classifier assigns both "location" and "organization" as the expected types.

2.2 Answer Extraction

We obtain a list of phrases from the top relevant documents. Before we can rank these phrases, we compute the following features of each phrase.

Frequency How many times the phrase appears in the top documents.

Overlap How many words appears in both the phrase and the question.

Length NSIR defines a longest length for each expected type of answer. If the length of the phrase, i.e., the number of words that the phrase contains, is less than or equal to the predefined length, then it gets 1 on this feature; otherwise, it gets zero.

Proximity This feature reflects how close the phrase is to the content (non "stop words") words in the question. The more question words the phrase is close to, and the closer the phrase is to the question words, then the higher score the phrase can get.

POSSIG Part-of-speech signature. Some types of questions are expecting answers of certain part-of-speech tag sequence. For instance, the answers to a *Number* question are usually tagged as CDs (numeral, cardinal). POS signature shows whether or not the phrase matches the expected POS tags.

LEXSIG Lexical signature. Some questions expect certain words, symbols, or patterns to appear in the answers. For example, phrases containing "percent" or "%" are more likely to be the answer to a *Percentage* question.

Word List We build a local database for both closed and semi-open categories, such as country names, currencies, universities, etc. A candidate phrase will get a bonus if it is contained in the corresponding local database. Say for a *Language* type question, the phrases such as "Arabic", "Chinese", "English", "Latin", etc., will be boosted by this feature.

Named-entity NSIR uses named-entity taggers to locate named-entities in open categories. The BBN Identifinder [2] is used when the target answer is person names, locations or organizations. A time-related named-entity tagger has been developed for the questions of the types such as $Date$, $Festival$, $Time$, etc.

Web ranking This feature is motivated by the vast amount of data available on the web. Exploiting the redundancy of the web data has appeared since TREC 2001 Question Answering Track (e.g., [3, 4]). To get this feature computed, NSIR runs questions first on a web search engine such as Google, then download the top web documents, computes the above features, and combines all these features. The final score for each phrase becomes the "web ranking" feature for the same phrase that is extracted locally.

2.3 Answer Ranking

Once the features of the potential answer phrases have been computed, NSIR is ready to rank the list of phrases using the linear combination of the features. Note that the ways to linearly combine the features vary. Different question types have different weights for each feature. Table 1 shows the weights of features for three example question types, namely, *Author*, *Language*, and $Year$ types. For example, as can be seen from the table, the *Named-entity* feature for *Language* type is 0 because no named-entity tagger is needed to answer *Language* type of questions. For another example, the Web *ranking* feature for $Year$ type has lower weight than the other two types, which means that NSIR does not exploit the redundancy of web data for $Year$ questions; this is because we are very likely to get recent years like "2002" which are not the intended answers of the TREC questions. Therefore, for each answer phrase, we can get a final score after combining the features in an appropriate way, then the phrase with the highest score becomes the top answer of the corresponding question. Until now, we have obtained *(qno, top answer, score)* for each question. The following part of this subsection will describe how we mark nil questions and rerank the answer list by confidence.

The corpus does not contain the answers to some questions, for which null responses will be marked as correct. Given that only one answer can be submitted for each question, returning a null response means that the top answer retrieved by our QA system will have to be given up. Our solution for marking nil questions is that any question whose top answer gets a score below a certain threshold, is marked as NIL. The threshold is learned by running NSIR on TREC-10 questions and observing a score so that the final scores of the NIL questions are below this score.

Our goal is to output the list of answers ranked by confidence. In our experiment, we use the final score of each top answer as the confidence level. Therefore, we rerank the answers by their final scores to get the intended ranking. However, for those NIL questions, we do need to find a way to boost their

Features	*Author*	*Language*	*Year*
Frequency	5	15	10
Overlap	-20	-20	-20
length	2	2	2
Proximity	15	5	5
POSSIG	3	3	0
LEXSIG	0	0	0
Word List	15	20	0
Named-entity	20	0	20
Web ranking	20	20	5

Table 1: Weights used in feature combination for question types *Author*, *Language*, and *Year*

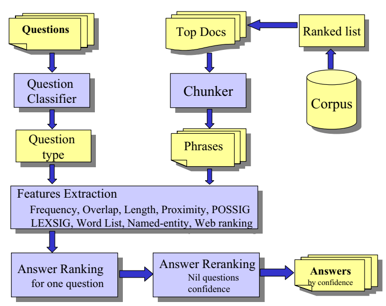

Figure 1: System Architecture of NSIR

rankings because they do not have a corresponding final score any more. Here we use a simple heuristic, namely, moving the first NIL question up to the 101st position, the second NIL question up to the 102nd position, and thereafter.

2.4 System Architecture

Figure 1 shows the architecture of the NSIR system. For each single question, we follow the steps below to extract the top answer.

a) Apply the question classifier to get the expected answer type (or types). This information will be used to compute the features for the candidate answers and rank the answers.

b) Obtain the top 20 documents according to the ranked list provided by

NIST; then apply LT-Chunk [5] on the top documents to get the phrases.

c) Once we get the expected answer type and the phrase list, we can start to compute the features for each phrase. Some features are independent of the expected answer type, such as $frequency$, $length$, etc. Other features are computed based on the answer type, such as $POSSIG$, $LEXSIG$, etc. To obtain the $Webranking$ feature, NSIR sends the current question as a query to Google and ranks the phrases that appear in the web documents, then uses the final scores as the $Webranking$ feature for the local phrases.

d) The steps above will produce a list of phrases with all the features computed. We linearly combine all the features with different weights predefined for the expected answer type. The candidate answer with the highest final score becomes the top answer to the question.

After NSIR gets the top answers along with their final scores for each question, it will apply to the answer list the heuristics for marking NIL questions and ranking answers by confidence that have been described in the last section. In the end, NSIR will output the answer list reranked according to the confidence levels.

2.5 Results and Discussion

We submitted three runs to this year's QA track. The main difference between them lies in the way of exploiting web data.

NSIRGN runs the questions on both local corpus and the web, using the results from the web as one more feature to rank the phrases that it gets from the local corpus.

NSIRG runs the questions only on the web, then locates the supporting local documents.

NSIRN runs the questions only on the local corpus without using the web.

Table 2 shows the official results on each of the three runs. NSIRGN outperformed the other two runs by about 6score, which suggests that the combination of local corpus and the web data performs better than using one of them alone.

In our experiment, we use the ranked list of documents provided by NIST, and run our system on the top 20 documents. Data shows that the top 20 documents contain answers for 311 out of the 500 questions. This is a major performance bottleneck for our system. More top documents should have been involved since the correct answers may be missed if the number of top documents is not enough. However, when we incorporate more top documents in the experiment, the number of candidate answers will be increased, which imposes difficulties on the later steps. Therefore, we believe that it is essential for a QA system to be able to rank high the right documents that support answering the input questions. If a large percentage of the top documents contain the correct

	NSIRGN	NSIRG	NSIRN
Wrong	379	388	404
Unsupported	18	16	7
Inexact	14	12	11
Right	89	84	78
Confidence-weighted Score	.283	.268	.269
# of NIL	175	195	63
Precision on NIL	.131	.128	.190
Recall on NIL	.500	.543	.261

Table 2: Official results on three runs of $NSIRGN$, $NSIRG$, and $NSIRN$

answers, then there will be much less noise when extracting answers, and much of the run time will also be saved.

One of the drawbacks in our system is that we did not effectively exploit the sentence level information. NSIR chunks the text into phrases right after it gets the top documents. Although the proximity feature carries some position information between the answer phrase and the question words, it is not able to distinguish between the cases where a phrase is always close to one of the question words and where a phrase appears near more question words. We plan to modify NSIR in the future so that it keeps the sentence level information when computing the proximity feature for the potential answer phrases.

There is another place that would improve the system performance. Recall the weights assigned to different features for each question type. Currently these weights are given manually by observing the system results on the previous TREC questions. Our question taxonomy contains so many classes, which makes it a hard task for humans to figure out the optimal weights for each question type. We plan to use machine learning techniques to learn these weights automatically.

This is our first time participating in the TREC QA evaluation (although the fourth author of this paper was earlier part of the IBM GuruQA project in TREC8 and TREC9 [6, **?**]). Though we didn't get very exciting results, we gained valuable experience for our future work on using the Web to improve question answering. We will continue our research using the TREC framework to get continuous improvements.

3 Using a multidocument summarizer for detecting new and relevant information in the news domain

3.1 Introduction and approach

This year, the Michigan team submitted five runs to the Novelty Track competition. We have based our first novelty detection systems on an extractive summarizer, MEAD [7] [8], which is currently under development at the University of Michigan. Ideally, existing multi-document summarization systems should already be choosing sentences from a cluster of related documents that are both relevant and novel in terms of their information content. Therefore, we were curious as to how MEAD would perform in the evaluation with just a few modifications. Since the instructions given to the judges indicated that the list of novel sentences must be a subset of the relevant sentences, and since we had limited resources, we first devoted our energies to training MEAD to detect relevant sentences. In future evaluations, we plan to focus on novelty detection in its own right.

3.2 The MEAD summarizer

MEAD compresses a cluster of topically-related documents into a summary of the user's desired length. It uses sentence features, such as length and position in the source document in order to rank the sentences as to their perceived importance to the document cluster. A third feature used by MEAD is the centroid, which measures the extent to which each sentence contains a set of key words that are important to the overall cluster.

Next, the sentences are ranked according to their combined score, which is a linear combination of all the sentence features used. Finally, MEAD has a reranker, in which relationships between sentences can be represented, and used to change the sentence rankings. For example, the default version of the reranking script attempts to prevent redundancy. It uses a cosine similarity metric to compare each candidate sentence (for inclusion in the summary) to each higher-ranking sentence. If the candidate sentence is too similar in terms of lexical context, it is penalized and is not included in the summary. Finally, the top remaining n-percent of the sentences (with the compression rate n being determined by the user), are returned to the user as the summary. For the relevant and novel sentence detection tasks, we used MEAD to produce extracts, which specify the top n-percent of the ranked sentences.

3.3 Observations from the Sample Data

Before looking at the sample data, we first divided it into a training and development set. The first step in developing a strategy was to examine the judges' lists of relevant and novel sentences in the four training clusters. We also ran

Cluster	Judge	Set
303	A	train
303	B	train
359	A	train
379	A	train
379	D	development
423	C	development

Table 3: Training and development sets

MEAD on these four clusters, in order to examine the sentence features of the sentences selected by the judges.

We found that a large proportion (approximately 75% for both the new and relevant lists) of the chosen sentences had relatively high centroid scores (between 0.35 and 0.7) as compared to those sentences that were not chosen by the judges. In addition, we found that judges tended to select groups of sentences with high centroid scores. Judges were not likely to chosen a given sentence in isolation. In other words, it was very common to find many groups of consecutive sentences included in the judges' lists. One MEAD feature that we felt had little correlation to novelty detection or to sentence relevancy was the position score. Judges did not seem more likely to choose sentences that are close to the beginning of the document over those that are further away.

3.4 Modifications to the MEAD summarizer

Based on our observations of the judges selected sentences in the four training clusters, we felt that the centroid score should be weighted heavily in the sentence ranking algorithm. Additionally, we felt that we might be able to model the tendency of judges to choose groups of sentence with high centroid scores in the MEAD reranker. In our modified reranking algorithm, if two or more consecutive sentences have centroid scores that are greater than the average score for all sentences in the cluster, a bonus is added to their scores.

Tables 4 through 7 show the precision and recall of the default version of MEAD, and that of MEAD with the new reranker. Unless stated otherwise, the compression rate used was 10%, since typically, this rate yielded the best result. Note that the default version of MEAD has centroid and position weights of 1, such that the initial sentence scores are calculated as $score(sentence) = centroid * 1 + position * 1$. The length feature has a cutoff value of 9 words. If a sentence is less than 9 words long, it is thrown out.

The modified reranker helped in almost all of the new and relevant sets and across all four sample clusters, and in no case did it do worse than the default MEAD. Therefore, we decided to incorporate it into this year's systems.

The next step was to develop new MEAD sentence features using the query data provided for each cluster of documents. There was a title, description, and

System	Recall	Precision
MEAD (relevant)	0.375	0.146
MEAD + new reranker (relevant)	0.500	0.200
MEAD (new)	0.400	0.146
MEAD + new reranker (new)	0.467	0.171

Table 4: Training cluster 303A

System	Recall	Precision
MEAD (relevant)	0.227	0.122
MEAD + new reranker (relevant)	0.227	0.122
MEAD (new)	0.333	0.122
MEAD + new reranker (new)	0.333	0.122

Table 5: Training cluster 303B

System	Recall	Precision
MEAD (relevant)	0.200	0.004
MEAD + new reranker (relevant)	0.250	0.007
MEAD (new)	0.166	0.040
MEAD + new reranker (new)	0.222	0.053

Table 6: Training cluster 359A

System	Recall	Precision
MEAD (relevant)	0.137	0.091
MEAD + new reranker (relevant)	0.196	0.122
MEAD (new)	0.146	0.091
MEAD + new reranker (new)	0.208	0.130

Table 7: Training cluster 379A

System	Description	Official average P*R	Corrected average P*R
Umich1	Centroid 1, Position 1, Length 9	0.019	0.026
Umich2	Centroid 20, Position 1, Length 15	0.016	0.016
Umich3	Centroid 20, Position 1, Length 15, 7%	0.011	0.012
Umich4	Centroid 20, Position 1, Length 15, 13%	0.019	0.019
Umich5	Centroid 1, Position 1, Length 9 Query-title-word-overlap 1	0.034	0.042

Table 8: Umich systems and results for finding relevant sentences

System	Description	Official average P*R	Corrected average P*R
Umich1	Centroid 1, Position 1, Length 9	0.017	0.023
Umich2	Centroid 20, Position 1, Length 15	0.014	0.015
Umich3	Centroid 20, Position 1, Length 15, 7%	0.010	0.011
Umich4	Centroid 20, Position 1, Length 15, 13%	0.017	0.018
Umich5	Centroid 1, Position 1, Length 9 Query-title-word-overlap 1	0.033	0.039

Table 9: Umich systems and results for finding new sentences

narrative associated with each cluster. We used these queries in implementing a word overlap feature for each. For example, in calculating the query-title-word-overlap feature score for a given sentence, if the title had four words and the sentence contained two of the four words, its query-title-word-overlap feature score would be 0.5. In our experiments with these features, we found the title query to be most useful. This is because the title of the cluster typically had many key words that were important to it, while the narratives and descriptions tended to resemble instructions to the judges on how to choose sentences relative to the topic. Therefore, they usually contained several sentences and many non-content words, and so their word overlap features were not as helpful in choose new and relevant sentences.

3.5 The Michigan systems and results

Table 8 gives an overview of the five systems we submitted to this year's evaluation, as well as the systems' average performance using the precision*recall metric. It should be noted that during our runs on the test data, we encountered problems related to the conversion of the SGML documents to XML for use with MEAD. This resulted in our systems not being able to process all 50 test clusters. Therefore, in the table we will include both the official evaluation results as well as the corrected results for all 50 clusters.

Our best system was Umich5, which used as one of its features the query-title-word overlap measure. This system greatly outperformed our other four systems in detecting both relevant and new sentences. Judging by the preliminary results sent out by NIST, the average performance of Umich5 seems to be somewhere in the middle of all the systems submitted this year.

3.6 Conclusions

It being the first year of the Novelty Detection Track, as well as our first time attempting novelty detection per se, we have learned many things from participating in the evaluation. One lesson learned about our back-end summarizer, MEAD, is that we must continue to make it more robust to the format of its text input files. As briefly mentioned previously, we encountered some problems with the conversion from SGML to XML during the testing that we did not see during our runs with the training data. We have since addressed this issue in our text cleaning scripts within MEAD.

We introduced a new feature within MEAD, query-title-word-overlap, which proved to be useful in system Umich5. However, we found that sentence overlap with the other query attributes, narrative and description of the clusters of documents, were not as helpful. We suspect this is because they appear to be more or less instructions to the judges about what to consider when choosing new and relevant sentences for the given cluster, and since they are longer than the titles and contain more function words, their overlap features do not really measure the extent to which a given sentence contains lexical items that are important to the overall cluster.

We have developed a new reranker within MEAD, which boosts the scores of groups of adjacent sentences that have large centroid scores, relative to the average centroid score for all of the sentences in the given cluster of documents. During our experiments with the training data, this seemed to model the tendency of judges to choose groups of consecutive sentences, rather than those in isolation. This reranker increased our precision and recall for both the new and relevant sentences sets for three out of four of the sample data clusters and left the fourth cluster's scores unchanged, when all other MEAD parameters were set at the default values. We might also try looking at using the query-title-word-overlap feature in this new reranker. However, in cases where the title is only a few words long, this might not work as well in attempting to boost scores of adjacent sentences, since one would expect authors to avoid using the same words in nearby sentences.

As is always the case, the judges' choices that we studied in the sample data while building our system seemed idiosyncratic at times, such that we could never obtain similar scores for multiple judges on a given cluster of documents. For example, for training cluster 303, we consistently got much higher precision and recall with judge A than with judge B. Therefore, in our training and experiments for next year's competition, we should use many different metrics in our self-evaluations, such as the intersection, union and mininum agreement of judges' new and relevant files.

Finally, for next year's competition, we will work on developing different strategies for detecting new versus relevant sentences. The new sentences are always a subset of the relevant ones. Therefore, we need to identify features that can distinguish the redundant sentences in the relevant lists that the judges are selecting out in order to create the file of new sentences. In short, this year we have seen how our summarizer, MEAD, performed on the novelty tasks with just a few simple modifications. Hopefully, by the next competition we will have identified more sophisticated features that we might implement, in order to enhance our system's performance.

References

[1] Dragomir R. Radev, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal. Probabilistic question answering from the web. In *The Eleventh International World Wide Web Conference*, Honolulu, Hawaii, May 2002.

[2] Daniel M. Bikel, Richard L. Schwartz, and Ralph M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231, 1999.

[3] Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. Data-intensive question answering. In *Proceedings of the Text Retrieval Conferences*, Nov 2001.

[4] C. L. A. Clarke, G. V.Cormack, T. R. Lynam, C. M. Li, and G. L. McLearn. Web reinforced question answering (multitext experiments for trec2001). In *Proceedings of the Text Retrieval Conferences*, Nov 2001.

[5] Language Technology Group. Lt chunk software. http://www.ltg.ed.ac.uk/software/chunk/.

[6] John Prager, Eric Brown, Anni Coden, and Dragomir Radev. Question-answering by predictive annotation. In *Proceedings, 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Athens, Greece, July 2000.

[7] Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *ANLP/NAACL Workshop on Summarization*, Seattle, WA, April 2000.

[8] Dragomir Radev, Simone Teufel, Horacio Saggion, Wai Lam, John Blitzer, Arda Çelebi, Hong Qi, Elliott Drabek, , and Danyu Liu. Evaluation of text summarization in a cross-lingual information retrieval framework. Technical report, Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD, June 2002.