THU QUANTA at TAC 2008 QA and RTE track

Fangtao Li, Zhicheng Zheng,Yang Tang, Fan Bu, Rong Ge, Xiaoyan Zhu, Xian Zhang, Minlie Huang

State Key Laboratory on Intelligent Technology and Systems,
Tsinghua National Laboratory of Intelligent Technology and Systems (LITS),
Department of Computer Science and Technology,
Tsinghua University, Beijing, 100084, China
lifangtao0923@163.com; zxy-dcs@tsinghua.edu.cn;

Abstract. This paper describes the systems of THU QUANTA in Text Analysis Conference (TAC) 2008. We participated in the Question Answering (QA) track, and the Recognizing Textual Entailment (RTE) track. For question answering track, we enhanced the traditional question answering system by sentiment lexicon based opinion analysis. The rigid list questions are divided into two categories based on their answer types. And two snippet extraction approaches are proposed to answer squishy list questions. For RTE track, we design different strategies to recognize true entailment and false entailment. The similarity between hypothesis and text is measured to recognize true entailment. We detect the exact entity and relation mismatch to recognize the false entailment. The evaluation results show that the proposed approaches are very effective for the QA and RTE tasks.

1 Introduction

In this year's Text Analysis Conference, we participated in two tracks: the Question Answering (QA) track and the Recognizing Textual Entailment (RTE) track. This paper reports on our two developed systems for these two tracks.

The TAC 2008 QA track is different from previous TREC QA. It focuses on finding answers to opinion questions. We develop the opinion question answering system by enhancing our past participated system QUANTA[3] with lexicon based sentiment analysis. We not only consider the topic relevance, but also pay attention to the sentiment match between question and answers. By analyzing the rigid list questions, we divide all the rigid list questions into two categories: the Opinion Holder Rigid List questions and Other Types Rigid List questions. Opinion Holder Rigid List questions refer to the questions whose answers are blog nicknames or blog holders. Other Types Rigid List questions refer to the questions whose answers are same to traditional factoid questions, such as person names, cities etc. The different strategies are designed for these two types of rigid list questions. We also propose two snippet extraction approaches to answer

squishy list questions. The evaluation results show that our methods are very effective. We rank 1st in this year's opinion question answering task.

This is our first participation on RTE track. We design a two-way strategy for this task. The main idea is that it is different to recognize true entailment and false entailment. Then we propose two separated approaches for recognizing true entailment and false entailment. For true entailment, we believe that the hypothesis can be transferred to the text by some transformation rules. This means that the text and hypothesis are similar in some levels of text linguistic representation. Therefore, we employ linguistic similarity to recognize the true entailment. For false entailment, we believe that there must be some mismatches between the text and the hypothesis. Therefore, we focus on "exact" entity and relation mismatch recognition to determine the false entailment. The "exact" means that the entity or relation plays a crucial rule in the textual entailment. If this "exact" mismatches, the false entailment can be determined. In this year's RTE track, the best result we achieved is 65.9% in accuracy.

2 Question Answering Track

Unlike past QA tracks, the TAC 2008 QA Track focuses on finding answers to short series of opinion questions. It includes two types of questions: rigid list questions and squishy list questions. We propose an opinion question answering framework by combining traditional topic based QA method and lexicon based sentiment analysis. Our QA system is augmented by sentimental lexicon based opinion analysis as follows: In question analysis stage, the questions are classified into two sentimental types, including positive and negative cases. After document retrieval, the sentimental determination is performed to filter out the sentiment mismatch documents. For snippet selection, we propose a unified model to combine the topic and the lexicon based sentiment to extract snippets with topic relevance and sentiment match. The official evaluation results show that our proposed approaches are effective. We achieved 0.156 for rigid list questions, 0.172 for squishy list questions and 0.168 for Average per-series score. Our rigid list question score and average per-series score both rank 1st among the 17 submit runs.

The architecture of our Question Answering system is similar to traditional question answering system. Several components are implemented in this framework: Blog Processing and Indexing, Question Analysis, Query Generation, Document Retrieval, and Answer Extraction for each type of question. The detailed introduction for each component and the official results are described as follows:

2.1 Blog Processing and Indexing

Since the Blog06 corpus is different from the AQUINT news corpus. It contains much noise. The blog corpus is tidied by several steps: filter bad chars directly; employ html

parser to extract the main text and eliminate the html tags; judge the char-set of the blog post and discard non-English documents. These works are quite essential for 2008 QA task, because blog data is the only corpus for this year's question answering track.

We use the public information retrieval tool Lucene to index the document corpus.

2.2 Question Analysis

We followed our past participated QA task to analyze the questions, including anaphora resolution, question normalization, question classification, and other syntactic and semantic analysis. In this year, for the opinion questions, the sentiment analysis is also employed:

1. Opinion holder, opinion object and opinion verb recognition.

We use a semantic role labeler (based on Propbank) to get all the predicates and corresponding Semantic roles from the parse tree of the normalized question. If a predicate is found in the semantic word list and has both semantic role A0 and semantic role A1, the predicate together with its A0 and A1 will be returned as opinion verb, opinion holder and opinion object respectively.

We also use some patterns (e.g . "reason for XXX", "opinion about XXX") to extract opinion objects.

2. Sentiment classification

All questions are classified into positive opinion question and negative opinion question categories based on their sentiment. We just identify the opinion words to finish this procedure. If the question contains "dislike", "negative" and other negative opinion words, we classify this question into negative opinion question category. If the question contains "like", "positive" and other positive opinion words, we classify this question into positive opinion question category. When the sentiment category is determined, we will just consider the same or similar category's documents and snippets for answer extraction.

2.3 Query Generation and Expansion

There are many sentence features can be used to generate queries, such as NP chunks, topic words, named entities and so on. The wikipedia's redirection function can be used to expand those features so as to find more relevant documents. After experiments on the TAC QA 2008 sample run data, we find that any single query generation strategy can hardly produce satisfactory results. The main reason is the questions are so different from each other. For some question, using the name entities as query can be very effective, for others, there could be no name entities found. A query generation strategy helps to find reasonable number of documents on some questions and no (or too many) documents on others. Meanwhile, the error of parser and NER and the noise caused by wiki also add uncertainty to query generation and expansion.

Considering robustness and efficiency, we generated a four-layer hierarchical query set for each question. The four query generation methods are listed as follows:

1) Only contain a single term------the question topic .
2) All the NP chunks, proper noun and digits having IDF score under a predefined threshold are collected as terms. All the terms are expanded through Wikipedia redirection data. The IDF scores are pre-computed by Google.
3) Named Entities are collected as terms. All the terms are expanded through wikipedia.
4) Named Entities are collected as terms without expansion.

2.4 Document Retrieval

Four queries are designed for each question. So after search, each question will have four returned document lists. We repeatedly collect the top N documents from these lists. That is to say, we collect first document from list 1, second from list 2…forth from list 4 and back to list 1…until we get N documents, or all the lists are empty.

In this year's QA TAC, we get the documents in two ways: the first gets top 500 documents by the designed method above. And the second is to include the top 1000 documents from official search results. We compute the union for these two methods.

2.5 Rigid list Answer Extraction

Since cross-topic opinion mining is quite complicated and difficult, answering all types of opinion questions in a unified structure is not reasonable and not possible.

In this point of view, we broadly divide rigid list questions into Opinion Holder Rigid List questions (OHRL questions in short) and Other Type Rigid List questions (OTRL questions in short) and solves these two types of questions using different schemes. The answer for OTRL question refers to blog nickname or blog holder, and the answer for OTRL question is the same as traditional factoid answers, including person name, book name, movie name and so on. If the questions ask for "blog", "blogger" or "who", this question belongs to the OHRL question, otherwise, it belongs to OTRL question. Furthermore, according to the target of the questions, OHRL questions can be divided into direct-OHRL questions and indirect-OHRL questions. Direct-OHRL questions aim to find opinion holders who directly express some opinions on a given topic, such as "Who are the people who enjoyed the movie "I Walk the Line"?", while indirect-OHRL questions focus on blogger names, like "What Bloggers expressed a positive attitude towards Mahmoud Ahmadinejad?" In another word, once someone's blog contains some relevant opinion about a given target, no matter it is copied or quoted, the blogger's name should be returned as an answer of the indirect-OHRL questions. Thus, answering indirect-OHRL questions is a bit easier than answering direct-OHRL questions. There are two kinds of answers of direct-OHRL questions. One is the author of an article and the other is

the speaker of a quoted statement. We will give more details about this in the following section.

Experiment results show that this "divide and conquer" method is very successful in solving rigid list opinion questions. We get top ranked F score at Rigid List Questions in 17 systems with OHRL Question F score to be 0.131 and OTRL F score 0.193.

2.5.1 Answer Extraction for Opinion Holder Rigid List questions

Our OHRL Question-Answering System consist several separate modules working in a sequential manner. In this way, we can easily share key modules in answering different kinds of questions and perform expensive operations on the dataset. In general, our QA system contains Question Processing Module, Document Retrieval Module, Answer Candidate Selection Module, Candidate Scoring & Resorting Module and Answer Generation Module. To accommodate the needs of OHRL Question-Answering, we adjust all modules except Question Processing and Document Retrieval Module.

1) Answer Snippet Candidate Selection Module

This module aims to locate relevant snippets from the retrieval results. As we know, when people express some opinion on a topic, the topic words ore relevant pronouns are usually very close to the opinion words. Besides that, opinion sentence on the same target always comes together. Based on this, we use a simple strategy to retrieve relevant snippets. For the first step, a document is split into sentences with topic words and pronouns marked. Then those sentences are grouped into several parts and sentences without any topic words or pronouns are removed. Then we score snippets by the occurrence of topics words and opinion words and select the top ranked snippets as the answer candidates.

2) Snippet Candidate Scoring & Re ranking Module

This is the cardinal module of the whole system. We compute several scores based on the occurrence of topic and opinion words.

a) title topic score = num of topic words in title / count of title words

b) title opinion score=num of opinion words in title * weight/ count of title words

c) snippet topic score = num of topic words in snippets / count of snippet words

d) snippet opinion score = num of opinion words in snippets * weight/ count of opinion words

We have tried several opinion dictionaries, including Hownet, Wordnet and Mpqa, to compute the opinion score and none of them works well because common opinion dictionaries contain too much noise compared to a given topic. For example, "big" expresses negative opinions when it comes together with "burden", though in most cases it is a positive word. To overcome this difficulty, we use a small opinion dictionary built by ourselves together with Hownet to compute the opinion score. We assign words of small opinion dictionary heavier weight to balance the accuracy and recall.

If a snippet's title topic score and title opinion score are higher than a specified threshold, this snippet is directly put into the answer pool. Otherwise, a final score is computed by:

*score(snippet) = a * title_topic_score + b * title_opinion_score +*
*c*snippet_topic_score + d * snippet_opinion_score* (1)

a, b, c, d are concluded from sample questions. We discard snippets whose score is less than a specified threshold. Then we select the top N ranked snippets into the answer pool.

3) Answer Generation Module

As we have mentioned above, there are three types of answers: bloggers, answers and text opinion holders. Since there are no available corpuses, it is impossible to train a statistic-based extractor to get opinion holders. So we rely on heuristic rules to extract exact answers. We write more than 30 regular expressions to extract blogger and authors from the ordinary html files. Besides that, we write 6 six simple rules to extract opinion holders. For example, we extract Tom as the opinion holder of the sentence "Tom said, ' I love it' " .

If the question is an indirect-OHRL question, we only have to extract the bloggers directly. To those direct-OHRL questions, we firstly extract opinion holders from nearby sentences. If we can't find any exact answers, we use the author-extractor to extract answers.

2.5.2 Answer Extraction for Other Type Rigid List questions

The answers for the Other Type Rigid List questions refer to the traditional factoid answers, such as person name, movie name etc. We process these questions by following steps:

1) Snippet Selection, we split the article into snippets with same number of sentences. And then, all snippets are scored according to BM25 function. If the snippet score exceed a predefined threshold, this snippet is extracted as candidate snippet.

2) Answer Candidate Generation: All candidate snippets are split into N grams as answer candidates. We mark the support snippet list for each candidate.

3) Candidate Filtering: According to different question, we filter candidates in different way. If we can easily access a name list from the external knowledge, such as movies/actors from IMDB, we use the list to filter the candidates. If the name list is not easy to fetch, we find the topic words and the answer type words for the question. We calculate the information distance between candidates and answer type under the condition of topic

$$d_C(x,y) = \frac{\log f(x,y,c) - \max\{\log f(x,c), \log f(y,c)\}}{\log f(c) - \min\{\log f(x,c), \log f(y,c)\}} \quad (2)$$

where x is candidate, y is answer type, c is topic, f(x) denotes the number of x[2]

Finally, we sort the candidate as a ranked list, and choose the top 50 candidates as the essential candidates set.

4) Sentiment Checking: for each essential candidate, we calculate the opinion score for its support texts based on sentiment lexicon Hownet. We count the number of opinion words near the candidate words. Here, we just consider the opinion words with the same polar to the question. We set different weights for opinion words based on the distance between opinion words and topic words. We calculate the opinion score for each support text. A threshold is defined according to the experiment in example questions, If the highest score of the candidate's support text is above this threshold, then we add the candidate to our final answer list, and the article, which contains the respond support text, is recognized as the support document.

2.5 Squishy List Answer Extraction

In this section, we describe our approaches to answer squishy list questions. We use two snippet extraction approaches to accomplish this task. One is the fixed number sentence based snippet extraction, and the other is question topic and its pronoun based snippet extraction.

2.6.1 Fixed Number Sentence based Snippet Extraction

The articles are split into snippets, which contain n (a fixed number, like n = 4) successive sentences with overlap. We will calculate the opinion score and topic score for each snippet:

1) Topic score

For topic score, we mainly consider two aspects: topic relevance score and informative score: the snippet must have much relationship with the topic, and the snippet should contain useful information. We use pattern and keyword based approach to estimate the topic relevance score. We construct several patterns, like definition patterns, why patterns, as features to rank the snippets. The keywords are extracted and expanded from the question words. We also collect some related lists as keywords, for example, given question “What reasons did people give for liking Ed Norton's movies?”, we collect all the Norton’s movie from IMDB as this question’s keywords. The position of keywords is also considered. If the keyword exists in the title or first sentence in the document, the topic score will be enhanced. To estimate the informative score, we calculate the average idf score for each snippet. The final topic score combines the topic relevance score and average IDF score in linear weight.

2) Opinion score

We design a sentiment lexicon based approach to compute the opinion score. Here the Hownet is employed as the sentiment lexicon. We first check the number of opinion

words existing in a predefined context window size. And then compute the snippet opinion score by adding opinion scores for all opinion words.

The final snippet score is computed by the following function:

$$final\_score(S) = \alpha * topic\_score(S) + \beta * opinion\_score(S) \quad (3)$$

We set a threshold to extract the top ranked snippets as the answers for each squishy list questions

2.6.2 Question Topics and Pronouns based Snippet Extraction

1). Topic words Extraction
We first construct the topic words, pronoun words for each question. The topic words are the topic of series questions and its expansions.
2). Pronoun Words Extraction
To get the pronoun words, we first classify each topic into group, female, male, and other categories. Group corresponds to pronoun words “they”, “them”, female to “her”, “she”, male to “he”, “him”, ‘his”, and other to “it”, “its”. Meanwhile, we extract the pronoun words from Wikipedia. For example, for topic “Nancy Grace”, the first sentence in Wikipedia is “Nancy Ann Grace (born October 23, 1959) is an American legal commentator, television host, and former prosecutor.” “commentator”, “host” and “prosecutor” are also extracted as pronoun words.
3). Opinion lexicon construction

As described above, we construct a small opinion lexicon by collecting the most used opinion words. For example, we collect the positive opinion words, including "good", "cool", "innovative", "wonderful", "great", "excellent", "amazing", "interesting", "like", "love", "pretty" etc, and negative words including “hate", "worse", "sad", "bother", "stupid", "angry", "bitch", "idiot", "annoying", "weird", "disgust", "disappoint", "frustrate", "sick", "nasty" etc. they are all common used words to express opinions.
4).Snippet Extraction

We first find the sentence, which contains both opinion word and topic word as the initial snippet. If the following sentence contain the topic words or pronoun words, the next sentence are also injected into the snippet. And the next sentence is recursively checked and injected. We will extracted all the snippet and rank them by the snippet’s length.

We submit two runs for squishy list questions: the first totally uses the first strategy, the fixed number sentence strategy. For the second run, we use the question topic and pronoun based sentence extraction approach for the question, whose topic is same as the question series, and other questions are processed by the fixed number sentence extraction strategy.

2.7 Evaluation Results

The evaluation results from the official evaluation from TAC 2008 are shown in the following table. Among 17 submitted runs, our results are competitive.

Table 1. the evaluation results for Quanta

	Best	Worst	median	Quanta1	Quanta2
Rigid	0.156	0.000	0.063	0.156	0.154
Squishy	0.186	0.018	0.091	0.136	0.172
per-series	0.168	0.011	0.093	0.149	0.168

3 Recognizing Textual Entailment Track

Textual entailment recognition task is to decide whether the text can entail the hypothesis. In this proposal, we propose a textual entailment recognition framework from two polarities. The assumption is that it is different to recognize the true entailment and false entailment. Therefore, different strategies are employed for True entailment recognition and False entailment recognition.

3.1 True Entailment Recognition

For the true entailment, we believe that the hypothesis can be transferred to the text by some transformation rules. There rules mentions all levels of linguistic analysis, including word transformation, phrase transformation, syntactic transformation and semantic transformation. Therefore, we compare the similarity between the hypothesis and the text in these levels of representation to recognize the True entailment:

1) Word match

The word similarity is calculated by the extended Local Lexical Matching method, enhanced by several WordNet relations.

2) Named Entity Match

Named Entity similarity is calculated for phrase transformation. Two kinds of Named Entity Recognition Tools, including Stanford NER and Sharp, are used to recognize eight types of Named Entities. And the Entities are extended by Wikipedia Redirection data set. We define five match relations: 1. Match; 2. Named Entity match, but Type not match (the NER tools' error) ; 3, Type Match, but Named Entity not match; 4, Mismatch; 5 Only appears in Hypothesis. We define different weight score for each match relation.

3) Syntactic Match

Two approaches are designed to compute the similarity in syntactic level. The first method is proposed based on the tree alignment approach [1], and we also consider the number of negation verbs. The second method calculates the path similarity in the syntactic tree.

4) Semantic Match

For semantic similarity, we first use a semantic role labeler to tag the predicate and all the args. We compare the verb similarity by WordNet distance, and then to recognize the augment similarity. The semantic match score is calculated as follows:

$$SRL\_score = predicate\_score * arg\_score \tag{4}$$

$$predicate\_score = \begin{cases} 1 & verb\ match \\ 0.5 & otherwise \end{cases} \tag{5}$$

$$arg\_score = \sqrt[n]{\prod_{i=1}^{n} Sim(H(i), \arg_i)} \tag{6}$$

Where n is the number of args and argi is the ith arg for predicate

5) Recognizing the true entailment

It is possible to set a threshold to recognize true entailment for each level similarity method. And we also use the machine learning tool weka to combine all the similarity values. The past RTE data sets are used to train a classifier to recognize the true entailment.

3.2 False Entailment Recognition

For the false entailment, we believe that there must be some mismatches between the text and the hypothesis. Therefore, we focus on "exact" entity and relation mismatch recognition to determine the false entailment. The "exact" means that the entity or relation plays a crucial rule in the textual entailment. If this "exact" mismatches, the false entailment can be determined. After similarity methods discussed in previous section, several levels of exact mismatch are used to detect false entailment, including word level, phrase level, sentence level, and syntactic relation level:

1) Number Mismatch

If the number appears in the Hypothesis, but doesn't appear in the Text, we predict that the entailment is false. For example, we predict #621 is false because the number "7.5" doesn't exist in Text. Meanwhile, we consider the number relations. For example, if the hypothesis contains "over 1000" and the text contains a numeric value above 1000, like 1024, we don't predict it as false entailment.

2) Time & Date Mismatch

Time & Date entity is a exact entity in RTE task. If the time and date mismatch, we predict it is false entailment.

3) Location Mismatch

For each location entity in H, if there is no corresponding entity in T, we predict it is false entailment. In some cases, one location can be expressed in different entities. Therefore we also use Wikipedia, country-nationality (China--Chinese) list and country-capital (China--Beijing) list to expand the location entities in T.

4) Quantifier Mismatch

We use the universal quantifiers (e.g. “all”, ”every”, ”each”) and negative words (e.g. “no”, ”none of”, ”never”) as exact entities. If a noun (or noun phrase) appears both in H and T and this noun (or noun phrase) is modified by universal quantifier (negative word) in H but not in T, we predict it is false entailment. The intuition behind this feature is that the constituents modified by universal quantifiers (e.g. “all”, ”every”, ”each”) or negative words are hard to satisfy.

5) “Say” relation mismatch

This relation mismatch means that somebody says something happens in Text, but in Hypothesis, it is said that somebody happens. Then we recognize this as false entailment.

6) “Locate” relation mismatch

For some special location description words, like “locate”, “base”, “from” exist in Text. We first align the objects, and then compare the subjects for these location markers in text and hypothesis. If the subjects mismatch, we predict it is false entailment.

7) Negation and subjunctive mismatch[5]

The text is first split into several small sentences. And then we compare each small sentence with hypothesis. If their similarity value is higher than a predefined threshold, we then check if only one has negation or subjunctive words. If only small sentence or only hypothesis has negation or subjunctive words, then we conclude negation and subjunctive mismatch, and predict it is a false entailment.

3.3 Submissions and Evaluation Results

Table 2. two-way evaluation results for Quanta

	Accuracy	Average precision
QUANTA1	0.659	0.6225
QUANTA2	0.623	0.5926

Table 3. three-way evaluation results for Quanta

	2-way Accuracy	3-way Accuracy	Avg precision
QUANTA	0.633	0.588	0.6332

We submit two results for two-way RTE task. The first submission QUANTA1 just uses the word match, named entity match and task description as features to recognize true entailment and uses all the mismatches to recognize false entailment. The second

submission QUANTA2 use all the match feature and task description features to recognize true entailment and uses all the mismatches to recognize false entailment. The training data set is the two-way task data in RTE 3. For the three-way RTE task, we just change the Negation and Subjunctive mismatch as "CONTRADICTION", and other false entailments are denoted as "UNKNOWN". The training data set is the annotated three-way task data in RTE 3.

4 Conclusion

In this paper, we describe our systems in TAC 2008 QA and RTE track. The question answering system is based on our past participated system, and enhanced by sentiment lexicon based opinion analysis. The Recognizing textual entailment system use different strategies to recognize true entailment and false entailment. The similarity methods are first employed to recognize true entailment and the exact entity and relation mismatch rules are then used to recognize false entailment. But both systems are still preliminary, there are many aspects to be improved.

Acknowledgement

This work is supported by National Natural Science Foundation of China (60572084, 60621062), National Basic Research Program of China (2007CB311003), and 985 key projects (SIST3002).

References

1. Marsi, E., Krahmer, E., Bosma, W., Theune, M. Normalized alignment of dependency trees for detecting textual entailment. In Proceedings of the Second PASCAL Challenges Workshop on Recognizing Textual Entailment, Venice, Italy. 2006
2 Xian Zhang, Yu Hao, Xiaoyan Zhu, and Ming Li. Information Distance from a Question to an Answer. In Proceedings of the 13th ACM SIG KDD conference, USA, 2007
3. Jianshu Sun, Xian Zhang, Fangtao Li, Xiaoyan Zhu. THUQA at TREC 2007 QA Track. In Notebook of 16th Text Retrieval Conference, Gaithersburg, USA, 2007
4. Rion Snow, Lucy Vanderwende and Arul Menezes. Effectively Using Syntax for Recognizing False Entailment. In Proceedings of the Human Language and Technology Conference of the North American Chapter of the ACL, pages 33-40, New York, June 2006.
5. Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, Bill Dolan. The Third PASCAL Recognizing Textual Entailment Challenge. in Proceedings of the Workshop on Textual Entailment and Paraphrasing, pages 1–9, Prague, June 2007