Answer Extraction from Passage Graph for Question Answering

Hong Sun[1] *, **Nan Duan**[2], **Yajuan Duan**[3], **Ming Zhou**[2]
[1] Tianjin University, No.92 Weijin Road, Tianjin, China
kaspersky@tju.edu.cn
[2] Microsoft Research Asia, No.5 Danling Street, Beijing, China
{nanduan, mingzhou}@microsoft.com
[3] University of Science and Technology of China, Hefei, China
dyj@mail.ustc.edu.cn

Abstract

In question answering, answer extraction aims to pin-point the exact answer from passages. However, most previous methods perform such extraction on each passage separately, without considering clues provided in other passages. This paper presents a novel approach to extract answers by fully leveraging connections among different passages. Specially, extraction is performed on a Passage Graph which is built by adding links upon multiple passages. Different passages are connected by linking words with the same stem. We use the factor graph as our model for answer extraction. Experimental results on multiple QA data sets demonstrate that our method significantly improves the performance of answer extraction.

1 Introduction

Question Answering (QA) is a task that aims to automatically give answers to questions described in natural language. Answer extraction is a key component in a QA system which generates the exact answer from the passages. Answer extraction first generates candidate answers from the passages and then rank them based on some scoring functions, for example, frequency of the candidates. Previous research has examined different methods of answer extraction such as Named Entity Recognition (NER) or pattern matching, however, they share a common property in that candidates are extracted separately from each passage, without considering any information provided by other passages.

However, clues provided by other passages can be useful. If we consider answer extraction as a task of classifying each word in the passages as being answer or not, the consensus information among all the passages can be useful when we perform the classification. Let's consider this from the basic lexical view: as all the passages are retrieved by the same query, it is possible that the same words in different passages are expressing the same meaning and as a result, they may all be answers, or none of them are answers. In addition, the more similar the contexts they share, the more likely their labels as answer candidates or not are the same. Thus it is possible to pin-point more occurrences of the correct answer or to correct the boundary during extraction.

Table 1: A TREC 2002 question. The passages are retrieved by using the question as a query

Question:	What is the state bird of Alaska?
Answer:	Willow Ptarmigan
Passages-1:	The official state bird of Alaska is Willow Ptarmigan
Passages-2:	This family of willow ptarmigan was filmed and photographed in...Park in Alaska
Passages-3:	Willow ptarmigan are fairly large birds, the size of a small chicken.

Consider an example illustrated in Table 1: the evidence in passage-1 is strong to indicate the phrase "Willow Ptarmigan" as a correct answer because the sentence structure is similar to the question and the phrase is capitalized which makes it easy to be identified. In contrast, the evidence in passage-2 and passage-3 are not so clear because the passages are stating other aspects about "Willow Ptarmigan". Further, "willow ptarmigan" in passage-2 is not capitalized, which makes it harder to be extracted. But if we leverage labels of "Willow Ptarmigan" in passage-1, the probability of labeling these two words as an answer candidate will be enhanced. In passage-3, "Willow" and "ptarmigan" are easily separated, but the boundary can be corrected by considering "Willow Ptarmigan" as candidate answer in passage-1. As a result, the frequency of the correct answer will be increased. This helps to increase the possibility for correctly answering the question.

Motivated by this observation, in this paper, we propose conducting answer extraction by fully leveraging connections among different passages. In particular, answer extraction is performed on a Passage Graph that is built upon all the passages retrieved for the same question. The connections among different passages are built by adding edges to the words with identical stems. In this way, whether a string can be identified as an answer candidate or not is decided jointly by two factors: (1)Evidence within the local passage (2)Evidence from labels of the same words in other passages. We adopt the factor graph as our model. In the factor graph, a random variable is introduced to each word within each passage to represent a 1/0 label. Among different passages, we add

*This work was finished while the author was visiting Microsoft Research Asia

a factor node connecting two variables whose corresponding words' stems are identical to represent the closeness between these variables. Experiments are performed on four public QA data sets. The results show that by leveraging relations among different passages, the performance of answer extraction can be significantly improved comparing with the method doesn't use such relations.

2 Related Work

Previous studies on answer extraction have discussed utilizing various structures for answer extraction, including patterns, named entities, n-grams and syntactic structures.

[Soubbotin, 2001] uses hand-crafted patterns to extract candidates from text for pre-defined question type (some work use question type to describe whether the question is asking about factoid, list, definition, etc. This paper focuses on factoid QA, we follow the notation in many QA papers which don't distinguish between question type and answer type). The scores of the candidates are determined by the patterns they come from. Without manual effort, [Ravichandran and Hovy, 2002] automatically learn such pattern sets and their scores. They send the question terms along with the answers to a search engine and extract patterns from the retrieved passages. [Ravichandran *et al.*, 2003] enrich the previous method by adding semantic types to the question terms, and use the automatically learned patterns as features to model the correctness of the extracted answers. Although it offers high precision, the pattern-based method for answer extraction is restricted by the pre-defined question type.

Beside patterns, different linguistic units are also extracted and ranked according to frequency. As commented by [Shen and Klakow, 2006; Chu-Carroll and Fan, 2011], most QA systems use Named Entity Recognition for answer extraction, such as [Prager *et al.*, 2000; Pasca and Harabagiu, 2001; Yang *et al.*, 2003; Xu *et al.*, 2003]. This method first extracts entities and then filters the list to retain candidates fixed in the expected answer type. Achieving a good performance usually requires implementing a named entity tool specifically for QA typology because many answer types are not covered by existing NER tools. However, developing such a recognizer is non-trivial and errors in determining the answer type will propagate to extraction. Another unit employed in answer extraction is n-gram. [Brill *et al.*, 2001] collect high frequent n-grams from documents retrieved from the web. The method then uses surface string features and hand-crafted patterns to determine the candidate types and performs filtering. Additionally, some of the text units are identified through external knowledge or a dictionary such as WordNet [Na *et al.*, 2002; Echihabi *et al.*, 2006], or title, anchor text and redirect metadata in Wikipedia [Chu-Carroll and Fan, 2011].

In addition, many methods rely on syntactic structures and extract noun phrases or dependency tree nodes from passages. These kinds of methods usually leverage similarity between question and answer sentences for ranking the candidates. [Sun *et al.*, 2005] decompose the dependency tree into triples and compute the similarity based on mutual information. [Shen *et al.*, 2005] use tree kernel function to compute the similarity and later explore the correlation of dependency paths between question and candidate sentences in [Shen and Klakow, 2006]. To overcome the surface gap between question expressions and candidate sentences, [Chrupala and Dinu, 2010] incorporate paraphrasing in mapping the dependency path and then rank the candidates based on language model.

The final method type is to view answer extraction as a term extraction process restricted by question [Sasaki, 2005b]. This method utilizes features coming from question, documents, and the matches between the two parts to tag words with *BIO* labels.

A factor graph [Kschischang *et al.*, 2001] is a bipartite graph defined over factors and variables that allows "global" functions to be factorized on different variables. Many NLP (Natural Language Processing) methods have discussed using factor graph to solve different problems such as relation extraction [Yao *et al.*, 2010], sentiment analysis [Täckström and McDonald, 2011] or event extraction [Liu *et al.*, 2012]. In this paper we employ factor graph as our model to perform answer extraction. Unlike previous methods of answer extraction, which separately extract answers in each passage, in this paper we perform answer extraction upon a graph built by linking all the passages related to the question together. This makes extraction not isolated among passages and allows the evidence provided by other passages to be fully leveraged.

3 Method

3.1 Task Definition

Given a question Q and its corresponding passages $P = \{p_1, p_2, ..., p_N\}$, the task is to pin-point the exact answer from P. Specially, we extract candidate c_i from each passage and later combine the same occurrences together to generate the candidate list $C = \{c_1, c_2, ..., c_k\}$. Here C is ranked base on frequency. Within each passage, we convert the extraction task into the classification on each word x_n^i in the passage p_n, with label $y_n^i \in \{1, 0\}$ to indicate whether it is an answer string or not. Here $n \in [1, N]$ is the serial number of the passage, i indicates the word is the i^{th} word in the passage. Next we link different passages to build the passage graph. Among different passages, we add edges $E_1 = \{e_{nm}^{ij}\}$ connected between the variables y_n^i and y_m^j, if x_n^i and x_m^j share the same stem (here we eliminate the links between stopwords and words in the question). The core task is to predicate $Y = \{y_n^i\}$ on the passage graph. After we get the predications, within each passage a word sequence with continuous labels 1 is extracted as a candidate to generate the candidate list C.

3.2 Model

We adopt factor graph as our model to resolve the predication on the graph. We formalize the factor graph as follows. Within each passage, we use f_n^i to denote the factor related to variable y_n^i. For each edge e_{nm}^{ij} across different passages, we use factor node f_{nm}^{ij} to factorize the features defined on this edge. Figure 1 presents an illustration to the factor graph, in which we assume the words x_1^3 and x_2^2 share the same lemma.

Given the question Q and the graph $G = (Y, F, E)$ where $Y = \{y_n^i\}$, $F = \{f_n^i\} \cup \{f_{nm}^{ij}\}$, E are edges connecting Y

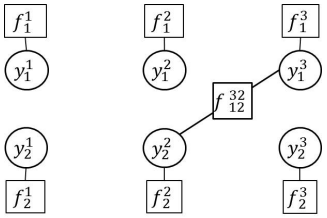

Figure 1: A factor graph for answer extraction. Circle nodes represent random variables for denoting word labels; rectangle nodes indicate factors connected to a single variable or two variables across different passages

and F. The task is to predicate Y on the factor graph, whose distribution is:

$$\begin{aligned} lnP(Y|G,Q) &= \sum_{n,i} lnf_n^i(y_n^i) \quad (1) \\ &+ \sum_{m,n,i,j,m\neq n} \delta_{nm}^{ij} \cdot lnf_{nm}^{ij}(y_n^i, y_m^j) \\ &- lnZ(G,Q) \end{aligned}$$

where $\delta_{nm}^{ij} = 1$ if x_n^i and x_m^j share the same stem and 0 otherwise. Here $f_n^i(y_n^i)$ is a factor node defined on variable y_n^i. $f_n^i(y_n^i)$ factorizes on multiple feature functions $h_{k_1}(y_n^i)$ defined on y_n^i:

$$f_n^i(y_n^i) = exp(\sum_{k_1} \lambda_{k_1} h_{k_1}(y_n^i)) \quad (2)$$

$f_{nm}^{ij}(y_n^i, y_m^j)$ is a factor node defined on y_n^i and y_m^j. $f_{nm}^{ij}(y_n^i, y_m^j)$ factorizes on feature functions $h_{k_2}(y_n^i, y_m^j)$ related to those two variables to model the closeness between them:

$$f_{nm}^{ij}(y_n^i, y_m^j) = exp(\sum_{k_2} \lambda_{k_2} h_{k_2}(y_n^i, y_m^j)) \quad (3)$$

$\{\lambda_{k_1}\}_{k_1=1}^{K_1}$ and $\{\lambda_{k_2}\}_{k_2=1}^{K_2}$ are their corresponding weights. Z is the normalization function which is given by:

$$\begin{aligned} Z(G,Q) &= \sum_Y \prod_{n,i} f_n^i(y_n^i) \cdot \\ &\prod_{m,n,i,j,m\neq n} f_{nm}^{ij}(y_n^i, y_m^j)^{\delta_{nm}^{ij}} \quad (4) \end{aligned}$$

Training process is to estimate parameters $\Lambda^* = \{\lambda_{k_1}\} \cup \{\lambda_{k_2}\}$ with:

$$\Lambda^* = arg\max lnP(Y|\Lambda, G, Q) \quad (5)$$

and then during the decoding process, the model predicates the labeling sequence Y^* on each passage based on:

$$Y^* = arg\max lnP(Y|\Lambda^*, G, Q) \quad (6)$$

In this work, we use loop belief propagation [Kevin P. Murphy and Jordan, 1999] for estimating the marginal probabilities in training and using L-BFGS to tune the parameters. For inferencing, max-product algorithm is used .Training and inferencing strategy follows the way mentioned in [Liu *et al.*, 2012]. Here we leave out the details for saving space.

3.3 Features

We define two feature sets $H_{graph} = \{h_{k_2}(y_m^i, y_n^j)\}_{k_2=1}^{K_2}$ and $H_{local} = \{h_{k_1}(y_m^i)\}_{k_1=1}^{K_1}$ to represent the two types of factor nodes in the graph. Both of the feature sets contain simple lexical features as well as complex features generated by POS (part of speech) tagging and dependency parsing. In the following, we illustrate the two sets.

Feature Set H_{graph}

This feature set depicts the probability that two words from different passages are conveying the same meaning. The higher the probability, the more likely the word labels will be the same. We model the probability mainly based on their contexts. Features of this set include: LCS of two passages containing the words; if the two words are both capitalized; if the two words are identical; if their POS tags are identical; if they have the same dependency label; if their dependency fathers are the same word; number of overlapping words in their dependency children nodes; number of the overlapping words in their surrounding word set (window size =5, similar hereafter); number of the words from other passages linked to the word.

Feature Set H_{local}

Features in this set are used to describe how likely a word is an answer within a given context. Unlike traditional features employed by NER, we add more features describing the relationship between the word and the question. These features come from four parts:

- **Question related features**: these features are defined to capture the extent that a given word is related to a question within certain context. This set includes: whether the word is in the question; whether the word's dependency father is in the question; whether it has a dependency relation with a main verb that occurs in the question; the number of its dependency children in the question; the number of the surrounded words in the question.
- **Lexical features**: these features describe the importance of each word, this category mainly includes traditional features used in NER: POS tag of the word and surrounding words; whether or not the word is capitalized; whether it is a stopword[1]; whether or not it contains a digit; whether it is punctuation; whether it is a person, location, or organization name.
- **Passage features**: these features capture the similarity between the passage and the question. They contain: the number of the passages' dependency triples matched with the question; LCS (Longest Common Sequence) matching between the question and the passage; the URL of the passage[2]; the title of the passage; the rank of the passage in the retrieval component.
- **Question features**: question type (person, location, etc.); question focus (who, when, where, etc.)

[1]The Stopword list used in our experiment can be found at http://norm.al/2009/04/14/list-of-english-stop-words/.

4 Experiments

In this section we describe the experiment settings and results, and perform analysis on the results.

4.1 Experiment Settings

QA Components: our QA system mainly follows the frame of a traditional QA pipeline and contains three parts:

- **Question analysis**: we use manually defined rules to determine the question type and focus. Question is used as the only query for retrieving passages.
- **Passage retrieval**: we use a well built search engine[3] to retrieve the passages and retain the top 10 snippets.
- **Answer extraction**: we select the candidate with the highest frequency as the final answer; if there's a tie, the most frequent candidates are ranked based on the maximum value of the score of their beginning words among the various passages they have been extracted from.

Text Preprocessing: snippets retrieved by the search engine are first broken into separate sentences. For each sentence in question or snippets, stemming, POS tagging, and dependency parsing are performed. We use the Stanford parser [Marneffe *et al.*, 2006] to generate the POS tag and dependency tree.

Data: We employ four QA data sets in our experiments:

- CLEF [Danilo *et al.*, 2008] 2007, 2008, 2011, and 2012 QA Data. We manually filter out non-factoid questions.
- Jeopardy![4]questions which has also been mentioned in [Chu-Carroll and Fan, 2011].
- NTCIR-5 [Sasaki, 2005a] English questions of English-Japanese cross lingual question answering task data.
- TREC [Voorhees, 1999] QA data set from the year 1999 to 2007. We add human effort to resolve the anaphora for some of the questions related to the given topic.

The data we use in our experiment is shown in Table 2. For training, we randomly select 2000 TREC QA data and use the remaining as testing data. Questions whose answers are missed by the previous step do not affect the performance when we evaluate the extraction component. So following the setting of [Shen and Klakow, 2006], we only retain questions whose answers are contained in the retrieved passages. Overall, this leave us around 60.26% of the questions. The binary recall (ratio of questions whose answers are contained in the search results) of our passage retrieval component is also shown in Table 2. The NTCIR data set achieves a low recall. This can be attributed to the fact that answers in this data set are dependent on the time or context the question are asked. We show the number of question types on the testing set in Table 3. Manual evaluation on 100 randomly selected questions (25 from each set) on testing set shows the accuracy of question type predication is 90%.

[2]We use passages retrieved from the Internet.

[3]We use the Microsoft Bing search engine http://www.bing.com/

[4]Jeopardy! (www.jeopardy.com) is an American quiz show.

Table 2: Training and testing data

Data	Data Set	Number	Retrieved Passages Contain Answer	Recall
Train	TREC	2000	1352	0.68
Test	CLEF	252	109	0.43
	Jeopardy!	2000	1130	0.57
	NTCIR	200	35	0.18
	TREC	835	560	0.67
	All	3287	1834	0.56

Table 3: Distribution of question types on test set

Type	Sub Type	Number
Named Entities	Person, Location and Organization	755
	Data and Number	215
Others	/	864

Evaluation: we evaluate the top k (k=1,5) number which is the number of questions whose answers are included in the top k candidates. We also report MRR [Voorhees, 1999]:

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank(ans_i)} \quad (7)$$

where $rank(ans_i)$ is the rank of the most top ranked correct answer of question q_i, N is the number of testing questions, only questions whose answers are covered by passage retrieval are counted.

4.2 Comparative Results

Compare with Method without Using Passage Graph

To testify the effect of the connections among passages, we construct the baseline by removing the factor nodes built upon variables across different passages. Thus, feature set H_{graph} and the factor nodes $\{f_{mn}^{ij}\}$ are eliminated. The training data and tool employed by this setting are identical to our proposed method. The baseline method is therefore similar to the one described in [Sasaki, 2005b] except that we add more dependency features and NER-style lexical features. Also, the Japanese POS features used in [Sasaki, 2005b] are absent.

The results are compared in Table 4[5]. By adding edges linked among different passages, all the metrics are better than the baseline's results. MRR in bold indicates a significant improvement with $p = 0.05$ (TREC's result is siginificant with $p = 0.1$). The improvement on NTCIR is not considerable, but the data set is very small so it cannot represent a statistical trend. Among all the data sets, the top 1 measure and the top 5 measure except in NTCIR are all improved. This result validates that involving relations among passages is effective for improving answer extraction. We also conduct reranking on baseline's results (denoted as Baseline-Rerank) based on 11 features (# of stopword, frequency, normalized

[5]The end-to-end pipeline gets 44% accuracy on TREC data(365 of 835) which makes the system comparable to the state-of-the-art TREC systems. However, such comparison is not fair because the corpus for retrieving answers are totally different.

Table 4: Experimental results of our method and removing the links between passages

Data Set	Method	Top 1	Top 5	MRR
CLEF	Baseline	51	63	0.520
	Our	61	69	**0.590**
Jeopardy!	Baseline	554	663	0.531
	Our	609	719	**0.582**
NTCIR	Baseline	16	27	0.603
	Our	18	26	0.613
TREC	Baseline	346	415	0.674
	Our	365	424	0.695
All	Baseline	967	1168	0.575
	Baseline-Rerank	981	1172	0.585
	Our	1053	1238	**0.618**

Table 5: Compare with NER on named entity questions

Method	Top 1	Top 5	MRR
NER	238	357	0.379
Our	529	610	**0.749**

unigram frequency around all candidates, answer type matching, length, etc.). Training process is carried out with *Ranking SVM* on TREC training data. The method is described in [Verberne *et al.*, 2009][6]. It appears that after reranking, the baseline result is improved, however, there's still some gap from the proposed method.

For efficiency comparison, as the passage-graph based method requires performing belief propagation, inferencing process takes 25ms to deal with each question compared with 5ms of the baseline. Time for extracting features for both of the methods are almost the same.

Compare with NER

Most work of answer extraction compares the performance within it's own research line, such as [Shen and Klakow, 2006] which compare their results with syntactic based methods. Our method follows the line using NER as a method for answer extraction but can be seen as a graphical model-based term extraction especially for the QA task. Therefore we compare the results with NER-based method. As it is not trivial to implement a sophisticated named entity recognizer designed specially for QA typology, we compare our method with an existing NER tool[7] on 755 *Person, Location and Organization* questions. Entities identified by the tool are kept and ranked based on frequency. The results are shown in Table 5, indicating that our method outperforms NER. This is because on the one hand, Stanford NER is trained on CoNLL data set that has a different context against search result documents. On the other hand, we use features related to questions that make our method more eligible the for the QA task.

Extraction and Ranking

Answer extraction usually first generates a list of candidates and then sorts them based on a ranking score. Some methods add more restrictions when generating the list, for example, NER-based method only retains candidates fixed in the question type. While others retrieve more candidates at first, and then rely on sophisticated typing or ranking functions, such as n-gram-based method or Wikipedia-based candidate generation policy described in [Chu-Carroll and Fan, 2011]. It is hard to say which kind of strategy is better. Our method belongs to the first kind. We restrict the generation process with manually designed features and obtain a high quality candidate list. One interesting question is whether by relaxing the restrictions in generation and later using these features in ranking, it is possible to achieve a better result?

Table 6: Compare with N-gram and Wikipedia Titles based extraction on test set excluding number and date questions

Method	Candidate Number	Top 1	Top 5	MRR
N-grams	652	242	592	0.250
N-grams After ranking	652	511	880	0.420
Wiki Title	39	419	710	0.340
Wiki Title After ranking	39	531	812	0.408
Our	7	927	1097	**0.619**

To answer this question, we conduct a heuristic experiment using n-grams (unigram, bigram and trigram) and Wikipedia titles as described in [Chu-Carroll and Fan, 2011] to generate candidates, followed by the same ranker used in Baseline-Rerank. To be fair to those two generation methods, 255 date and number questions are excluded. Results are shown in Table 6. Before reranking, frequency is used to rank the candidates (so does our method as illustrated before). Both of the methods' results are improved after re-ranking. However, the performance is still not very satisfactory. This is because the more candidates a method generates, the more stress there will be on the ranker and makes ranking more difficult. Achieving a better result requires more perfection on the ranking function as well as ranking feature selection, which are not parts of our discussion.

4.3 Result Analysis

Analysis of the Passage Graph Effect

We further perform some manual analysis on the positive cases to study how the graph improves the performance against the baseline method. It turns out the graph takes effect from two perspectives. First, if the baseline method has identified the answer, adding the connections may further enhance the answers' frequency by correcting the boundary or pin-pointing more occurrences of the answer. This has been illustrated in the example in Section 1. Second, if the baseline method misses the answer, our method may retrieve it by considering the global information, thus improving answer coverage. In our observation, the binary recall of answer extraction on the test set is improved from 65.53% to 68.59%. Consider the example in Table 7. The baseline method without considering passage relations doesn't extract the correct answer. All the passages containing the answer are similar

[6] SVM^{Rank}: www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

[7] We use Stanford NER open source toolkit.

Table 7: An example to show top 3 candidates given by our method and the baseline method

Question: What was the profession of American patriot Paul Revere? **Answer**: silversmith **Passages**: Paul Revere, ..., and silversmith.	
Our Method	Baseline Method
silversmith	Independence
Born	Boston
Independence	Huguenot

Table 8: Results of different question types

Type	Method	Top 1	Top 5	MRR
Named Entity	Baseline	643	756	0.714
	Our	655	751	0.719
Others	Baseline	324	412	0.420
	Our	398	487	**0.505**

to the passage in the table which has a long distance dependency relation between the answer and key words in question which makes question related scores are lower. Scores of lexical features are also low, so evidence within each passage is not strong enough to support the string as an answer. However, if we look at the global context, *silversmith* has a high frequency under similar contexts, this returns a high score of the second part of equation (1) over label 1 and enables the extractor to identify it as a candidate.

Results for Different Question Types

The results of different question types are shown in Table 8. The results show that our method contributes more to questions asking about non-named entities. Such questions make up a large portion of the questions and are more difficult to deal with. Our method achieves comparable results for named entity questions because the baseline method can efficiently identify named entities by, for example, leveraging capitalization or dictionary features. However, if the answers are not named entities, they are not easily identified. Under such conditions, by leveraging information among all the passages, our method is more effective to extract the answer.

Feature Set

We compare effectiveness of different feature sets in Table 9. One feature set in H_{local} is removed at one time and we measure the decay of the performance. We view MRR as the main measure as it captures the average rank of the answer. The results show that, question related features are most important. They measure how much each word is related to the question. Those features are especially necessary for questions that don't ask about named entities. In addition, question feature set is the second important. Question features are identical to all the words for the same question, it plays a role in adjusting the threshold when performing the classification. For example, the weight of "Location" over label 1 is negative, so it requires a larger score of label 1 for a word to be determined as a candidate. The lexical features are helpful to extract most of the named entities from passages. Without these features, low frequency named entities, especially dates and numbers will be missed. Passage features play the least important role, we induce the reason as these features do not directly depict the quality of each candidate and they're equivalent to every candidate within the same passage.

Table 9: Results when remove different feature sets

Setting	Top 1	Top 5	MRR
-Question Related	1015	1128	0.579
-Question	1026	1141	0.587
-Lexical	1039	1157	0.595
-Passage	1048	1158	0.600
Use All	1053	1238	0.618

Table 10: Distribution of cause of error

All	Wrong	Extraction Miss	Ranking Incorrect
1834	781	620	161

Error Analysis

Errors of our method come from two parts: one is when the answer is missed in the candidate list, the other is caused when the answer is extracted but not ranked correctly. The distribution of the errors on test set is shown in Table 10. The results show that main error is the extraction miss. To further analyze the errors, we randomly select 150 wrong cases from the testing set (15 from NTCIR and 35 from the other three sets respectively) and manually check the reason. Among them our method provides correct answers to 45 questions, but they're judged as wrong because the candidate does not match the surface form of the given answer. Besides, there are 35 questions our method answers with a wrong boundary. In addition there are many cases in which answers are partially given and most of these answers are compound nouns. 46 answers are missed by our method, 20 of them are neither named entities nor capitalized and appear only once, which makes them very difficult to be identified.

5 Conclusion and Future Work

In this paper, we propose to leverage relations among different passages for answer extraction. We perform extraction on a Passage Graph which is built upon all the passages related to the same question. Edges across different passages are built by linking words with the same stem. Extraction is achieved by giving each word a label of *1/0* to indicate it is an answer string or not. Factor graph is adopted to inference the feature weights. Experiment results show that by adding links among different passages, the performance of answer extraction is significantly improved.

One current limitation of our method is that we only consider connections among words with the same stem. However, semantic equivalents are also important. We'll add such information to further improve the performance of our method in the future. Also, impact from multiple queries issued by the same question deserve further study.

References

[Brill *et al.*, 2001] Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. Data-intensive question answering. In *TREC*, pages 393–400, 2001.

[Chrupala and Dinu, 2010] Grzegorz Chrupala and Georgiana Dinu. Enriched syntax-based meaning representation for answer extraction. In *Proceedings of SIGIR Workshop*, 2010.

[Chu-Carroll and Fan, 2011] Jennifer Chu-Carroll and James Fan. Leveraging wikipedia characteristics for search and candidate generation in question answering. In *Proceedings of AAAI*, 2011.

[Danilo *et al.*, 2008] Giampiccolo Danilo, Forner Pamela, Herrera Jesús, Pe nas Anselmo, Ayache Christelle, Forascu Corina, Jijkoun Valentin, Osenova Petya, Rocha Paulo, Sacaleanu Bogdan, and Sutcliffe Richard. Advances in multilingual and multimodal information retrieval. pages 200–236, 2008.

[Echihabi *et al.*, 2006] Abdessamad Echihabi, Ulf Hermjakob, Eduard Hovy, Daniel Marcu, Eric Melz, and Deepak Ravichandran. *How to Select Answer String*. Springer Netherlands, 2006.

[Kevin P. Murphy and Jordan, 1999] Yair Weiss Kevin P. Murphy and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of Uncertainty in AI*, pages 467–475, 1999.

[Kschischang *et al.*, 2001] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[Liu *et al.*, 2012] Xiaohua Liu, Xiangyang Zhou, Zhongyang Fu, Furu Wei, and Ming Zhou. Exacting social events for tweets using a factor graph. In *Proceedings of AAAI*, 2012.

[Marneffe *et al.*, 2006] Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, 2006.

[Na *et al.*, 2002] Seung-Hoon Na, In-Su Kang, Sang-Yool Lee, and Jong-Hyeok Lee. Question answering approach using a wordnet-based answer type taxonomy. In *Proceedings of TREC*, 2002.

[Pasca and Harabagiu, 2001] Marius A. Pasca and Sanda M. Harabagiu. High performance question answering. In *Proceedings of SIGIR*, pages 366–374, 2001.

[Prager *et al.*, 2000] John Prager, Eric Brown, Anni Coden, and Dragomir Radev. Question-answering by predictive annotation. In *Proceedings of SIGIR*, pages 184–191, 2000.

[Ravichandran and Hovy, 2002] Deepak Ravichandran and Eduard H. Hovy. Learning surface text patterns for a question answering system. In *Proceedings of ACL*, pages 41–47, 2002.

[Ravichandran *et al.*, 2003] Deepak Ravichandran, Abraham Ittycheriah, and Salim Roukos. Automatic derivation of surface text patterns for a maximum entropy based question answering system. In *Proceedings of HLT-NAACL*, 2003.

[Sasaki, 2005a] Yutaka Sasaki. Overview of the ntcir-5 cross-lingual question answering task. In *Proceedings of NTCIR-5 Workshop Meeting, December 6-9, 2005, Tokyo, Japan*, 2005.

[Sasaki, 2005b] Yutaka Sasaki. Question answering as question-biased term extraction: a new approach toward multilingual qa. In *Proceedings of ACL*, pages 215–222, 2005.

[Shen and Klakow, 2006] Dan Shen and Dietrich Klakow. Exploring correlation of dependency relation paths for answer extraction. In *Proceedings of ACL*, 2006.

[Shen *et al.*, 2005] Dan Shen, Geert-Jan M. Kruijff, and Dietrich Klakow. Exploring syntactic relation patterns for question answering. In *Proceedings of IJCNLP*, pages 507–518, 2005.

[Soubbotin, 2001] M. M. Soubbotin. Patterns of potential answer expressions as clues to the right answers. In *Proceedings of TREC*, pages 293–302, 2001.

[Sun *et al.*, 2005] Renxu Sun, Hang Cui, Keya Li, Min-Yen Kan, and Tat-Seng Chua. Dependency relation matching for answer selection. In *Proceedings of SIGIR*, pages 651–652, 2005.

[Täckström and McDonald, 2011] Oscar Täckström and Ryan T. McDonald. Semi-supervised latent variable models for sentence-level sentiment analysis. In *Proceedings of ACL (Short Papers)*, pages 569–574, 2011.

[Verberne *et al.*, 2009] Suzan Verberne, Clst Ru Nijmegen, Hans Van Halteren, Clst Ru Nijmegen, Daphne Theijssen, Ru Nijmegen, Stephan Raaijmakers, Lou Boves, and Clst Ru Nijmegen. Learning to rank qa data. evaluating machine learning techniques for ranking answers to why-questions. In *Proceedings of theWorkshop on Learning to Rank for Information Retrieval (LR4IR) at SIGIR*, 2009.

[Voorhees, 1999] Ellen M. Voorhees. The trec-8 question answering track evaluation. In *Proceedings of TREC*, 1999.

[Xu *et al.*, 2003] Jinxi Xu, Ana Licuanan, Jonathan May, Scott Miller, and Ralph Weischedel. Answer selection and confidence estimation. In *AAAi Spring Symposium on New Directions in QA*, 2003.

[Yang *et al.*, 2003] Hui Yang, Hang Cui, Mstislav Maslennikov, Long Qiu, Min yen Kan, and Tat seng Chua. Qualifier in trec-12 qa main task. In *Proceedings of TREC*, 2003.

[Yao *et al.*, 2010] Limin Yao, Sebastian Riedel, and Andrew McCallum. Collective cross-document relation extraction without labelled data. In *Proceedings of EMNLP*, 2010.