MIG at NTCIR-11: Using Lexical, Syntactic, and Semantic Features for the RITE-VAL Tasks

Po-Cheng Lin, Shu Yu Lin, Chih Kai Haung, Chao-Lin Liu
Department of Computer Science, National Chengchi University
64 Chih-Nan Road Section 2, Wen-Shan, Taipei 11605, Taiwan
{101753028, 102753020, 102753029, chaolin}@nccu.edu.tw

ABSTRACT

In this paper, we describe our methods for the English and Chinese RITE-VAL tasks. We extracted relevant sentences from Wikipedia to verify the correctness of the query statements. Computational models that considered various linguistic features were built to select Wikipedia articles that contained these relevant sentences. We adopt Linearly Weighted Functions (LWFs) to balance the importance of every features and judge the answer of each query statement by the outputs of LWFs.

Team Name

MIG

Subtasks

RTIE-VAL Fact Validation EN/CS/CT

General terms

Algorithm, Experimentation, Languages

Keywords

Natural Language Processing, Textual Entailment, Information Retrieval, Negation and Antonyms, Linearly Weighted Functions

1. INTRODUCTION

Applications of natural language processing techniques have sprung up in the recent years. To improve the performance, experts extract useful linguistic features from words, sentences, and even articles, trying to figure out the meanings of texts and speeches. If we can let computer know natural languages' meanings, it will be helpful for teaching or even creating a more convenient tools.

In RITE-VAL, the purpose is to determine whether a query statement is contradiction or entailment[13]. Contradiction means that the query statement is not true. In practice, we judge this by searching the Wikipedia [11]. If we find clues that go against the query statement, we would classify the statement as contradiction. In contrast, if we find clues that support the query statement, then the sentence will be classified as entailment. At last, if there is no obvious reasons to support to disapprove the sentence we tag the sentence as "Unknown".

We try to find relatively important words in query statement. By finding such information, we can extract some related articles and sentences in Wikipedia. After that, we employ some features to compute the weight for each of related sentence, and consider some semantic features for the final judgments. We also adopt the data provided by NTCIR to train the parameters and threshold for our LWFs. If the score for a query statement is higher than the threshold, then the sentence is "Entailment", otherwise the sentence will be "Contradiction".

We explain our methods for extracting articles and sentences that are potentially related to a given query from Wikipedia in Section 2. We will then deliberate on further processing steps for the extracted materials in an attempt to sift those information that are really related to the query in Section 3. In Section 4, we discuss our methods for entailment recognition.

2. EXTRACT ARTICLES AND SENTENCES

In this section, we adopt two ways to extract related articles and sentences for English corpus and Chinese corpus respectively, we will show as below.

For English corpus, as the picture in Figure 1, we disassemble query statement into word, phrase and strings. Extract those synonyms by accessing WordNet[12]. Then, we combine all the results we get as keywords to do request to Wikipedia. If there is a article title in Wikipedia that is match our keywords then we will extract whole article as our related articles.

After that, we split all the articles into sentences. However, not entire sentences are related to the query statement, so we make a filter to choose the sentences those are relevant to query statement. Our strategy is to choose sentences those have greater than or equal to 2 words match to the query statement, and those words' parts of speech must be noun, verb or adjective. Finally we can shrink the range into some

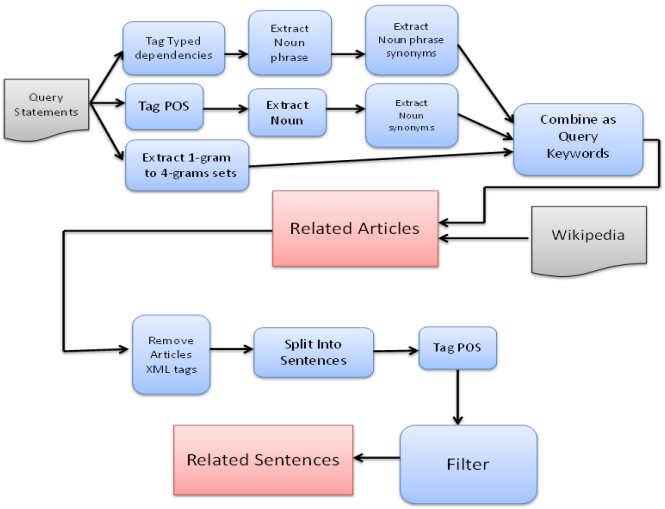

Figure 1.The flow of extracting article and sentences for English corpus.

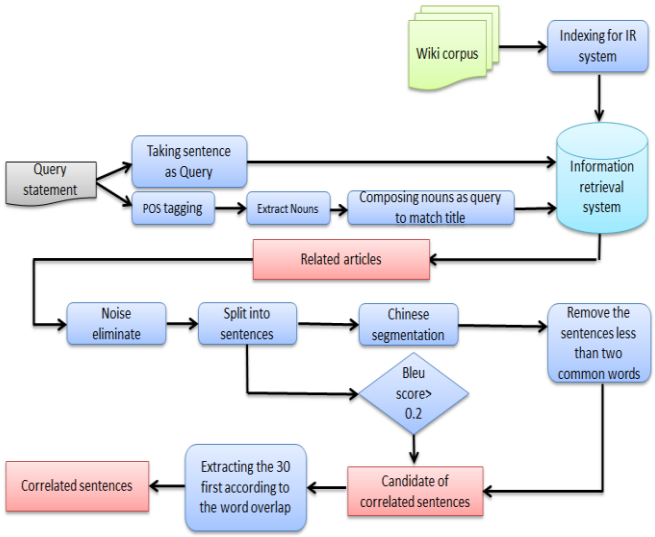

Figure 2. The flow of extracting article and sentences for Chinese corpus.

sentences those are more related to query statement.

For Chinese corpus, as the picture shown Figure 2, the approaches of retrieve correlated atricles from wikipedia corpus have two ways.

One way is using wikipedia corpus to set a retrival system by Lucene[6]. Then, using the sentences as query to retrieve articles. The other is extract Nouns in sentences. Checking whether there are titles same as Nouns or not. If they are same, we also make it as correlated article. Thus, we split article's paragraphs to sentences. After that we use the word overlapped and Bilingual Evaluation Understudy (BLEU)[1] measurement to calculate the correlation between argument and sentences in the articles, and take top 30 as related sentences.

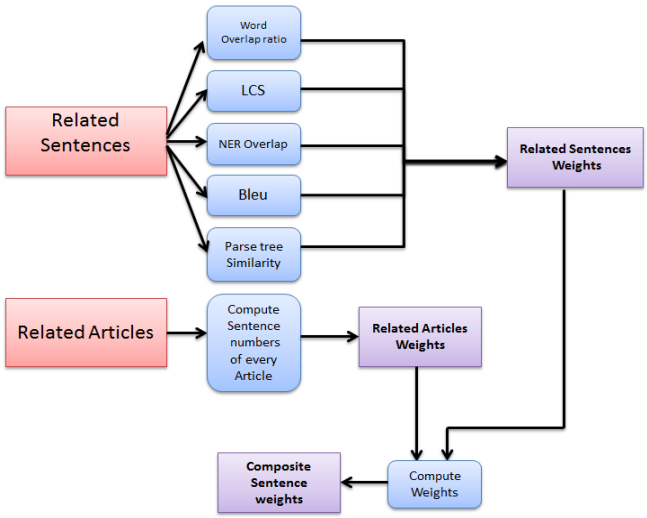

Figure 3. The method of calculating sentence relatedness.

3. COMPUTE SENTENCE RELATEDNESS

After section 2, every query statement can obtain their related sentences from Wikipedia's article. But there is a question inside, how much relatedness between related sentence and input words. The degree of relatedness will greatly effect the recognition result, so we try to quantify the relatedness degree of each related sentences .

In Figure 3, sentence relatedness is compose into two parts. One is related sentence itself, the other is the article where related sentence comes from. In the first part, we extract five features value between related and query statement.

A. Word Overlap Ratio :

Comparing query statement and related sentence. If there exist same words in both sentences, then we call it word overlap. Therefore, the more same words extist, the more ratio it will get.

$$\text{Word Overlap Ratio} = \frac{\text{Word Overlap Number}}{\text{Word Number Of Query statement}} \quad (1)$$

B. Longest Common String Similarity(LCSS):

We extract query statement and related sentence's Longest Common String (LCS)[5] as an important linguistic feauture, that is, if we have two sentence below :

Sentence1: "ABCDEFGH"
Sentence2: "ABIJDEFKLMN"

Then, the LCS will be "ABDEF". And according to the formula (2), we can compute that the LCSS in this example is 0.375.

$$\text{LCSS} = \frac{\text{LCS}}{\text{Length of Query Statement}} \quad (2)$$

C. Name Entity Overlap Ratio :

We tag name entity[7] (PERSON, ORGANIZATION, LOCATION) on both query statement and related sentence by using Stanford NER. After that, we compare both sentence's name entity, The more same name entity exist, the higher ratio it will get.

D. Bilingual Evaluation Understudy :

BLEU is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. We use it to check two sentence's similarity.

E. Parse Tree Similarity :

In order to obtain sentence semantic structure, we use Stanford Parser[8] to parse query statement and related sentence. To extract sentence's semantic structure similarity[3] there is a procedure as below:

Step1:Compute LCS between related sentence and query statement.

Step2:Compare the difference between the LCS and query statement, if there exist a token that LCS is different from query statement, then we record it into a table.

Step3:We do insertion operation to insert some node into the LCS by cheking out the table. After the LCS becomes query statement, the node number we add is the cost of insertion operation. Therefore, if insertion operation is less, then it shows that the pair of sentence have less differences. After the procedure, we can obtain a value that can represent Parse Tree Similarity.

To figure out related sentence weights, we set the features value above as a, b, c, d and e. We make them as a set of space vector. Therefore, after extracting the features, we can assume the starting point is (0,0,0,0,0) and corresponding related sentences coordinates are (a,b,c,d,e). Hence, we can compute two sentences distance value. The higher value it get, the higher sentence weights it will obtain.

The second part is related article weight. If we can find many related sentences in a article, then we can also say, that article have certain degree of representation. Therefore, we try to figure out the importance of each related article as below's formula, and the related sentences' weight will also be decided depends on each articles they come from. Let x = Total related sentence numbers of each relate articles; y = Total related sentence numbers of each query statement.

$$\text{Related Article Weight} = \frac{x}{y} \quad (3)$$

Finally, we multiplied sentence weight and article weight as every sentence's relatedness. And that value is represent that how related between query statement and related sentence. We will use it in our following section. Let S_w = Related sentence weight; A_w = Related article weight.

$$\text{Sentence Relatedness} = S_w \times A_w \quad (4)$$

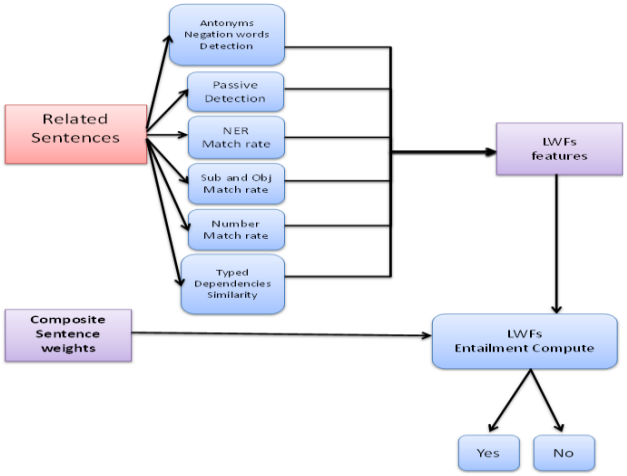

Figure 4. The method of entailment recognition.

4. ENTAILMENT RECOGNITION

In this section, we use some semantic features to figure out the relations between related sentence and query statement. As the Figure 4, we also adopt Linearly Weighted Functions(LWFs)[4] to be our main method to validate our sentence.

Here comes the introduction of each features :

A.Antonyms, Negation Adverb and Negation Words Detection :

To determine a pair of sentence which is entailment or contradiction, antonyms, negation adverb and negation words are important information to deduce it. In other words, even though two sentences have high value of Word Overlap Ratio, if there exist antonyms or negation words, the meaning will be entirely different.

We compare query statement and related sentence. If there exist same words that is modified by antonyms, negation adverb or negation words, then we compute the relation that is positive or negative. If we get the relation is positive then the feature's value is 1, else the value will be -1.

B.Passive Detection :

We compare query statement and related sentence. Detecing both sentences those are active or passive sentence. If those are the same, then we give 1 as feature's value, else the value will be -1.

C.Subject and Object Match Ratio:

If there have same subject or object in query statement and related sentence, In other words, we can say both two

sentences are probably discribing same things or showing same information. As the formula below, the more same subject or object we get, the higher feature value we can get.

$$\text{Subject and Object Match Ratio} = \frac{\text{Subject and Object Overlap Number}}{\text{Number Of Subject and Object in Query statement}} \quad (5)$$

D. Number Match Ratio :

Number is important information to detect two sentences' relationship. As the formula shown below, if two sentences have totaly same number, then they have high posibility as a positive entailment relationship.

$$\text{Number Match Ratio} = \frac{\text{Number word Overlap Quantity}}{\text{Quantity Of Number word in Query statement}} \quad (6)$$

E. Typed Dependencies Similarity:

Typed Dependencies[9] means the direct relationship between two words. We make two matrixes to put two sentence typed dependency sets inside. Then we multiply itself twice, three and four times to find its transition matrix. After that, we can obtain the matrix that appears the result of two, three and four steps of typed denpencies relations, it means, we have the indirect relationship between every words in every sentences. Therefore we can compare both related sentence and query statement's transition matrixes. If there exist same typed denpencies relations then we can say those two typed dependencies are somehow interconnected to each other.

After extracting the features above, we combine those features and all related sentences' weight. Putting those value into Linearly Weighted Functions as below. Hence we can train every feature's parameter(α , β, γ, δ, ε) and observe that which linguistic feature is much useful to help us validate the sentence.

Linearly Weighted Functions(LWFs) is the formula as below. We assume one query statement has 30 related sentences and F1 to F5 are the features we introduce above. Moreover, α , β, γ, δ, ε are parameters of each features and w_S , w_A are sentence weight and article weight respectively.

$$\begin{aligned}\text{LWFs Score} = {} & w_{S1}{*}w_{A1}\,(\alpha\ F1_{S1} + \beta\ F2_{S1} + \gamma\ F3_{S1} + \delta\ F4_{S1} + \varepsilon\ F5_{S1}) + \\ & w_{S2}{*}w_{A2}\,(\alpha\ F1_{S2} + \beta\ F2_{S2} + \gamma\ F3_{S2} + \delta\ F4_{S2} + \varepsilon\ F5_{S2}) + \\ & \cdots \\ & \cdots \\ & w_{S30}{*}w_{A30}\,(\alpha\ F1_{S30} + \beta\ F2_{S30} + \gamma\ F3_{S30} + \delta\ F4_{S30} + \varepsilon\ F5_{S30}) \end{aligned} \quad (7)$$

We can obtain LWFs Score by the LWFs formula and validate the query statement whether it is "Entailment", "Contradiction" or "Unknown" as formula (8). Besides, the value of Threshold1 and Threshold1 are trained by RITE-2 and RITE-VAL training data, so we validate our query statement according to the value of threshold1 and threshold2. Finally if LWFs Score >= Threshold1, then the sentence will be tagged into "Entailment". If LWFs Score <= Threshold2, then the sentence will be tagged into "Contradiction". If Threshold2< LWFs Score < Threshold1, then the sentence will be tagged into "Unknown". Let Threshold 1 = t1; Threshold 2 = t2.

Table 1. Formal run result

Task	Accuracy	Macro-F1
RITE-VAL EN Run 01	51.06	49.41
RITE-VAL EN Run 02	50.53	49.64
RITE-VAL CN Run 01	36.70	30.88
RITE-VAL CN Run 02	35.89	31.07
RITE-VAL CS Run 01	36.70	30.88
RITE-VAL CS Run 02	35.89	31.07

$$\textit{Query Statement Recognition} = \begin{cases} Entailment & LWFs\ Score >= t1 \\ Contradiction & LWFs\ Score <= t2 \\ Unknown & t2 < LWFs\ Score < t1 \end{cases} \quad (8)$$

5. EVALUATION

The table 1 shows the formal run results of RITE-VAL "EN", "CN", and "CS" tasks. The parameter sets of each run, we pick better performance set in training data. In RITE-VAL EN task, we can see the accuracy and macro-f1 falls on about 50%. It is not as good as our anticipation, but we get the second place in the task.

In RITE-VAL CN and CS tasks, our best performance is 36.7% of accuracy and 30.88% of marco-f1. In our training data performance, we can get about 48% of accuracy. The reason why we get large of difference between training set and formal run set is, the data's differences is large. The training data's content, is describe about common sense knowledge, the difficulties is about the test of elementary school. And the formal run data is much harder than training data, so that is the main reason for causing the differences.

6. DISCUSSION

This paper reports our system in the NTCIR-11 RITE-VAL EN, CT, CS sub-tasks. The results were not as good as we expected, and we thought that there are three possible ways to make our performance better.

First is about features, we should build more features concerning semantic level, that can help us know more about meaning. Second, we should try different ways to filter our related sentences, that is, a more considerate procedure is in need to identify those really relevant sentences in Wikipedia.

Currently, we chose linearly weighted functions to be our method to judge the answer. We should try more advanced

methods like SVM[10] for the task. We probably should compare different methods, so that we may have a better performance.

Acknowledgment

This work was supported in part by the grant MOST-102-2420-H-004-054-MY2 from the Ministry of Science and Technology of Taiwan.

References

[1] BLEU, http://en.wikipedia.org/wiki/BLEU

[2] Wei-Jie Huang and Chao-Lin Liu. 2013. NCCU-MIG at NTCIR-10: Using Lexical, Syntactic, and Semantic Features for the RITE Tasks. In Proceedings of the 10th NTCIR Conference, Tokyo, Japan, 430-434.

[3] Guoliang Li, Xuhui Liu, Jianhua Feng, and Lizhu Zhou, Efficient Similarity Search for Tree-Structured Data, Author Affiliations: Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China

[4] Linearly Weighted Functions, http://en.wikipedia.org/wiki/Weight_function

[5] Longest Common String, http://en.wikipedia.org/wiki/Longest_common_substring_problem

[6] Lucene, http://lucene.apache.org/core/

[7] Stanford NER, http://nlp.stanford.edu/ner/

[8] Stanford Parser, http://nlp.stanford.edu/software/lex-parser.shtml

[9] Stanford Typed Dependencies, http://nlp.stanford.edu/software/dependencies_manual.pdf

[10] SVM, http://en.wikipedia.org/wiki/Support_vector_machine

[11] Wikipedia, http://en.wikipedia.org/wiki/Main_Page

[12] WordNet, http://wordnet.princeton.edu/

[13] Yotaro Watanabe, Yusuke Miyao, Junta Mizuno, Tomohide Shibata, Hiroshi Kanayama, C.-W. Lee, C.-J. Lin, Shuming Shi, Teruko Mitamura, Noriko Kando, Hideki Shima and Kohichi Takeda. 2013. Overview of the Recognizing Inference in Text (RITE-2) at the NTCIR-10 Conference, Proceedings of NTCIR-10 Conference.