Recognizing Textual Entailment using Lexical, Syntactical, and Semantic Information

Yann-Huei Lee
Institue of Information Science, Academic Sinica
128 Academia Road, Section2
NanKang, Taipei 115, Taiwan
+886-2-27883799 ext 1560
andycyrus@gmail.com

Shafqat Virk
Institue of Information Science, Academic Sinica
128 Academia Road, Section2
NanKang, Taipei 115, Taiwan
+886-2-27883799 ext 2415
virk.shafqat@gmail.com

Lun-Wei Ku
Institue of Information Science, Academic Sinica
128 Academia Road, Section2
NanKang, Taipei 115, Taiwan
+886-2-27883799 ext 1808
lwku@iis.sinica.edu.tw

ABSTRACT

This paper describes our system for participating in the system validation subtask of NTCIR-11 RITE-VAL. We trained a SVM model with LibSVM using features extracted from labeled sentence pairs. Besides features based on lexical, syntactic and semantic analysis, we introduce a novel approach of extracting "concepts" from a sentence and generating features based on it. Unlabeled testing sentence pairs' features are extracted through the same process, and the SVM model that we have trained predicts their labels.

Keywords

RITE, Concept-Based system, Entailment Recognition

Team Name

ASNLP.

Subtasks

System Validation (Chinese-Simplified, Chinese-Tradition).

1. INTRODUCTION

RITE-VAL [1] is a generic benchmark task that addresses major text understanding needs in various NLP/Information Access research areas. It evaluates systems that detect entailment, paraphrase, and contradiction in texts written in Japanese, English, Simplified Chinese, or Traditional Chinese. Entailment is a classic logic problem in artificial intelligence. In the RITE task it becomes a natural language processing problem, where the experimental materials are texts.

We participated in the system validation subtask in this year's RITE-VAL, which requires participants to develop systems that could classify the entailment relation between a sentence pair formed by a text (T) sentence and hypothesis (H) sentence. There are two subtasks, including binary-classification (BC) and multi-classification (MC). Given the pair of sentences (T, H), the BC subtask is to determine whether T entails H, a label "Y" is given if entailment exists, and "N" if not. The MC subtask is to determine the entailment direction or contradiction. The labels used in BC subtask are "Y" and "N". The labels defined in MC subtask are "F" (for forward entailment, which T entails H), "B" (for bidirectional entailment, which T and H entails each other), "C" (for contradiction), and "I" (for independence).

The rest of this paper is organized as follow. In Section 2, we introduce the architecture of our entailment system. In section 3 we describe our system in detail, including preprocessing and feature extractions. In section 4 we present the evaluation results from the official formal run. In section 5 we summarize our experiment.

2. SYSTEM ARCHITECTURE

The overall architecture of our entailment system is shown in Figure 1. Our system contains a preprocessing model, a concept extraction model, a feature extraction model and a SVM model. The procedure is as follow:

1. The text (T) and hypothesis (H) is first put through a preprocessing model including the following processes:

   A. Traditional Chinese to simplified Chinese conversion.

   B. Numerical transfer.

   C. Generate segmentation, parse tree, dependency tree through Stanford parser.

   D. Segmentation and POS tagging by CKIP parser.

2. Parse tree of each sentence is then put through the concept extraction model to extract concepts from each sentence.

3. Calculate features from each text and hypothesis pair to generate a feature vector for each sentence pair.

4. Use LibSVM [8] to train a model with feature vectors generated from training data sentence pairs.

5. Use LibSVM and the trained model to predict classification of the formal data.

3. SYSTEM DESCRIPTION

In this section, we describe each component of our system, including preprocessing, concept extraction, feature calculation, and concept feature calculation.

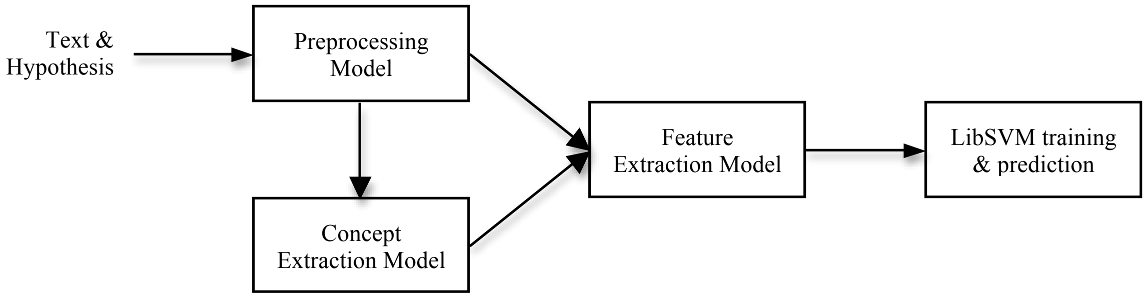

Figure 1. System Architecture

PREPROCESSING

In this subsection, we explain the preprocessing functions done before each text and hypothesis pair is ready for feature extraction. These preprocesses include: simplified Chinese transformation, numeric format transformation, segmentation, parse tree generation, dependency tree generation, and CKIP parsing.

3.1.1 Simplified Chinese Transformation

Our system rely on public tools like Stanford parser for Chinese segmentation, parse tree generation, and dependency tree generation. The Stanford parser has a better performance with simplified Chinese. Therefore, we need to convert traditional Chinese into simplified Chinese.

We convert traditional Chinese characters to their simplified Chinese characters by a Java Chinese Convertor [2].

3.1.2 Numeric Format Transformation

Numbers in Chinese sentence could be represented in Chinese numerals or Arabic numerals or a mix of both forms. Different numeric formats would lead to different interpretation of the sentence. In order to confront this issue, we capture certain strings within a sentence, base on certain rules, and convert Chinese numerals into Arabic numerals. Figure 2 shows an example of the transformation.

Original: Top Gear 在全世界拥有多达 3 亿 5,000 万观众。(Top Gear has more than 350,000,000 audiences in the world)

Transformed: Top Gear 在全世界拥有多达 350000000 观众 。

Figure 2. Numeric Format Transformation Example

3.1.3 Stanford Parser

After each sentence has been transformed to simplified Chinese and unified numeral formats, each sentence is first segmented through the Stanford Word Segmenter [3], which is a necessary process for identifying words within a Chinese sentence.

The segmented sentence is then used by the Stanford parser [4] to generate a context-free phrase structure grammar representation (parse tree) and a typed-dependency representation (dependency tree).

3.1.4 CKIP Parser

Besides Stanford parser, we also use the CKIP parser [5], a system designed for Traditional Chinese, to get segmentation and part-of-speech tag information.

3.2 CONCEPT EXTRACTION

We introduce a novel idea of decomposing natural language text to smaller semantic expressions called "concepts", which includes continuous or dis-continuous parts of a sentence that makes some semantic sense. [7]

To generate concepts from a sentence, we first need a structured representation (i.e. a parse tree) using Stanford parser. This is achieved in section 3.1.3.

Next, the resulting parse tree is semantically analyzed to identify and label semantic arguments associated with the verbal predicate of a sentence. For this purpose, we built a semantic role labeling (SRL) system using Penn TreeBank and Propbank data. A novel semantic role classification approach is used to incorporate contextual information, while labeling the semantic arguments of a predicate.

Figure 3 shows an example output of our concept extraction tool. It shows the predicates and arguments extracted from the sentence. Pairing up every argument with its semantic role and predicate, we can form a "concept".

Input Sentence:
《红楼梦》被评为中国古典章回小说的最高杰作，思想价值和艺术价值极高。
("Dream of Red Chamber" was rated as the highest masterpiece of Chinese classical novels; it's highly valued in ideological and artistic.)

Semantic Role Labeling:

Predicate	Argument	SRL
极高	思想价值和艺术价值	ARG0
评为	中国古典章回小说的最高杰作	ARG1
	《红楼梦》	ARG0

Extracted Concepts:

极高 - 思想价值和艺术价值(ARG0)
评为 -《红楼梦》(ARG0)
评为 - 中国古典章回小说的最高杰作(ARG1)

Figure 3. Concept Extraction Example

3.3 FEATURE CALCULATION

In this subsection, we describe in detail the features that are used to generate a feature vector for each text (T) and hypothesis (H) pair, which would be used for training and prediction with LibSVM. All features return a value between 0 to 1, which is done by normalizing the result with certain measures of T or H. This normalization process could also reflect the direction of entailment, which is important in MC. All features are calculated in aspects of T and H to reflect direction of entailment as well.

3.3.1 Exclusive Word

This feature calculates the amount of words in H that does not exist in T. This amount is then normalized by the amount of words within T or H.

3.3.2 Sentence Length Based Entailment

This feature return 1 if H has more words than T. Return 0 otherwise.

Another version is to return 1 if T has more words than H, return 0 otherwise.

3.3.3 N-Gram Overlap

This feature first calculates the amount of identical n-grams within T and H on character level. Then normalize it with the amount of possible n-grams that could contain T or H as the result for this feature. We consider only bigrams and trigrams.

3.3.4 Word Overlap

This feature is identical to n-gram overlap, but we consider n-grams at word level.

3.3.5 Matching Words

This feature first calculates the amount of identical words within T and H. Then it is normalize it by the amount of words in T or the amount of words in H.

3.3.6 LCS Similarity

This feature first calculates the length of the longest common string (LCS) of T and H. Then it is normalized by the total length of all possible substrings in T or in H.

3.3.7 Levenshtein Distance

Levenshtein distance, also know as edit distance, considers the minimum amount of operations for one string to transform to another. Here, we calculate the amount of operations needed to transform from T to H at word level. Then it is normalized by the total amount of words within T or H.

3.3.8 Common String Overlap

This feature first calculates the total length of all common substrings in T and H. Then it is normalized by the length of T or H .

3.3.9 Cosine Similarity

This feature first builds a word vector for T and H. Each dimension of the vector represents the occurrence frequency of a certain word within the sentence. We then calculate the cosine similarity between the word vector of T and H.

3.3.10 Disconnected Dependencies

The dependency relation generated by Stanford parser is in the format as follow:

$$Relation\ (\ Head\ -\ Head\#\ ,\ Modifier\ -\ Modifier\#\ )$$

In this feature we first extract all nouns (words with POS tag of "NN", "NR", "NT" labeled by Stanford parser) from T and H to get $NounList_T$ and $NounList_H$.

In the aspect of H, all nouns in $NounList_H$ have to exist in $NounList_T$ for further calculation. Then we go through all dependencies in H, which both *Head* and *Modifier* is in $NounList_H$, and calculate the amount of dependencies in T that has the same relation but different *Head* and *Modifier*. This amount is then normalized by the amount of dependency in T or H.

As for the aspect of T, all nouns in $NounList_T$ have to exist in $NounList_H$ for further calculation. Then we go through all dependencies in T, which its *Head* and *Modifier* is in $NounList_T$, and calculate the amount of dependencies in H that has the same relation but different *Head* and *Modifier.* This amount is then normalized by the amount of dependency in T or H.

A non-strict version of this feature is to neglect the criteria that all nouns in $NounList_H$ have to exist in $NounList_T$, or all nouns in $NounList_T$ have to exist in $NounList_H$.

3.3.11 Missing Arguments

In this feature we first extract all nouns (words with POS tag of "NN", "NR", "NT" labeled by Stanford parser) from T and H to get $NounList_T$ and $NounList_H$.

In the aspect of H, we calculate the amount of nouns in $NounList_H$ that is not in $NounList_T$. This amount is normalized by the amount of nouns in $NounList_T$ or in $NounList_H$.

As for the aspect of T, we calculate the amount of nouns in $NounList_T$ that is not in $NounList_H$ This amount is normalized by the amount of nouns in $NounList_T$ or in $NounList_H$.

3.3.12 Missing Relation

In this feature, we go through all dependencies in H and check if there's any dependency in T that has the same "relation". If no dependency of the same "relation" exists in T, we have a missing relation. We calculate the amount of missing relations in H and normalize it by the amount of dependencies in T or H.

In the aspect of T, we would go through all dependencies in T and calculate the amount of missing relations, and then normalize it by the amount of dependencies in T or H.

3.3.13 Connected Dependencies

In this feature we first extract all nouns (words with POS tag of "NN", "NR", "NT" labeled by Stanford parser) from T and H to get $NounList_T$ and $NounList_H$.

In the aspect of H, all nouns in $NounList_H$ have to exist in $NounList_T$ for further calculation. Then we go through all dependencies in H, which its *Head* and *Modifier* is in $NounList_H$, and calculate the amount of identical dependencies in T. This amount is then normalized by the amount of dependency in T or H.

As for the aspect of T, all nouns in $NounList_T$ have to exist in $NounList_H$ for further calculation. Then we go through all dependencies in T, which its *Head* and *Modifier* is in $NounList_T$, and calculate the amount of identical dependencies in H. This amount is then normalized by the amount of dependency in T or H.

A non-strict version of this feature is to neglect the criteria that all nouns in $NounList_H$ have to exist in $NounList_T$, or all nouns in $NounList_T$ have to exist in $NounList_H$.

3.3.14 Independent Negative Dependency

In this feature, we return 1 when there exist a dependency in H that satisfy,

1. The relation of the dependency is “neg”.
2. In the dependencies of T exists one that has the same *Head* as the current dependency in H and its relation is not “neg”.

0 is returned if such case doesn’t exist.

Another version is considering from the aspect of T, which we return 1 when there exist a dependency in T that satisfy,

1. The relation of the dependency is “neg”.
2. In the dependencies of H exists one that has the same *Head* as the current dependency in T and its relation is not “neg”.

0 is returned if such case doesn’t exist.

3.3.15 Contradiction Dependency

In this feature, we return 1 when there exist a dependency in H that satisfy,

1. The relation of the dependency is “neg”.
2. In the dependencies of T doesn’t exists one that has the same *Head* as the current dependency in H and its relation is “neg”.

0 is returned if such case doesn’t exist.

Another version is considering from the aspect of T, which we return 1 when there exist a dependency in T that satisfy,

1. The relation of the dependency is “neg”.
2. In the dependencies of H doesn’t exists one that has the same *Head* as the current dependency in T and its relation is “neg”.

0 is returned if such case doesn’t exist.

3.3.16 CKIP POS tag

CKIP part-of-speech tags give more detail labeling in Chinese for proper nouns, places, location, time, numbers, and amounts with 9 tags: “Nb”, “Nc”, “Ncd”, “Nd”, “Neu”, “Neqa”, “Neqb”, “Nf”, “Ng”, “Nh”. If H doesn’t contain or have different content than T under these labels, it is likely that T doesn’t entail H.

So we generate a feature value for each tag. Each feature checks if T contains the tag in concern, -1 is returned when no words in T is labeled with the tag. Extract all words that are labeled with the tag in concern in T and H to get two word lists: WordList$_{T,\ tag}$ and WordList$_{M,\ tag}$. If WordList$_{T,\ tag}$ and WordList$_{M,\ tag}$ matches each other in every element return 1, return 0 if else.

3.3.17 CILIN Synonyms

The features described above matches words base on exact match, which would consider synonyms as different entities and lead to misjudging when calculating features to determine entailment.

To encounter this issue, we applied a Chinese thesaurus tong2yi4ci2ci2lin2 ("同義詞詞林") [6], which lists 65,464 words in the format as shown in Figure 4.

加快(speed up)	Ih08.01.001.%
加速(speed up)	Ih08.01.002.%
放慢(slow down)	Ih08.02.001.%
降速(slow down)	Ih08.02.002.%
減速(slow down)	Ih08.02.003.%
	…
推遲(delay)	Ih07.02.001.%
延遲(delay)	Ih07.02.002.%

Figure 4. Chinese thesaurus tong2yi4ci2ci2lin2 data example

Each word is given a unique code in the format as follow,

MainClass# . SubClass1# . SubClass2# .%

In Figure 3, we can see the first two words “加快” and “加速” both means “speed up”, they have identical MainClass# and SubClass1# but different SubClass2#. So is the following three words “放慢”, “降速”, and “減速”, all means “slow down”.

So during certain feature calculations which involves matching two words, beside exacting string match we also check and see if both words can be found within this thesaurus and have the same MainClass# and SubClass1#. If this criterion is satisfied, we consider the two words to be identical as well. This is applied in the following features mention previously to form a new feature for calculation: Exclusive words, disconnected dependencies, connected dependencies, and missing arguments.

3.4 CONCEPT FEATURE CALCULATION

In this subsection, we discuss about the features calculated from concepts extracted from each T and H.

Considering concepts represent information extracted from each sentence, T is likely to entail H if T has the same or more information than H. And it’s likely that entailment doesn’t exist if H contains information not included in T. The following features tend to reflect this information in different aspects.

3.4.1 Concept Amount Difference

This feature calculates the difference of the amount of concepts in H compared to T. This amount is then normalized by the amount of concepts in T or H.

Another version is to calculate the difference of the amount of concepts in T compared to H, and then normalize by the amount of concepts in T or H.

3.4.2 Predicate Miss

Since concepts are based on predicates extracted from a sentence, amount of difference in predicates would reflect the magnitude of difference in two sentences.

In this feature, we calculate the amount of predicates in H that is not in T, and normalize by the amount of predicates in H.

Another version is to calculate the amount of predicates in T that is not in H, and normalize by the amount of predicates in T.

3.4.3 Argument Miss

Beside predicates, argument is also an important component of concepts.

So in this feature we look for the amount of arguments in H that is not in the arguments of T under the same predicate. This amount is then normalized by the total amount of arguments in H.

Another version is to calculate the amount of arguments in T that is not in the arguments of H under the same predicate. This amount is then normalized by the total amount of arguments in T.

3.4.4 Argument SRL Miss

Since each argument has been given a semantic role label, it is likely that a pair of sentence has the same information, if the argument from the same predicate with the same SRL is identical.

So in this feature we look for the amount of arguments in H that is not in the arguments of T, under the same predicate and same SRL. This amount is then normalized by the total amount of arguments in H.

Another version is to calculate the amount of arguments in T that is not in the arguments of H under the same predicate and same SRL. This amount is then normalized by the total amount of arguments in T.

3.4.5 SRL Miss Amount

Semantic role labels of “ARGM-LOC” and “ARGM-TMP” are labels for location and time. It is likely that T entails H when the location and time information in H is identical to the same information in T.

So in this feature, we go through all concepts in H to see if we could find arguments with SRL as “ARGM-LOC” or “ARGM-TMP”, and if there exists an identical argument in T with the same SRL. If such argument doesn’t exist in T, a SRL miss occur. We calculate the amount of SRL miss and normalize it by the amount of arguments in H with SRL as “ARGM-LOC” or “ARGM-TMP”.

Another version would be going through concepts in T and calculate the amount of SRL miss of H, then normalize it by the amount of arguments in T with SRL as “ARGM-LOC” or “ARGM-TMP”.

4. EVALUATION

The Training data for BC task are 1321 labeled sentence pairs in Traditional Chinese from RITE2, and 581 labeled sentence pairs each in Traditional and Simplified Chinese from RITE-VAL training data. As for the MC task, we have 1321 labeled sentence pairs in Traditional Chinese from RITE2, and 581 labeled sentence pairs each in Traditional and Simplified Chinese from RITE-VAL training data.

The test data for Traditional Chinese and Simplified Chinese in BC and MC each have 1200 unlabeled sentence pairs.

Our system is developed with knowledge sources and tools that handle Simplified Chinese, so when dealing with Traditional Chinese sentence pairs in training and testing data, we convert them into Simplified Chinese, as mentioned in 3.1.1, before further processing.

We submitted one run for each task. The official evaluation results are shown in Table 1~4.

Table 1. Evaluation result on binary-class (BC) classification in Simplified Chinese (CS) data set

	Precision	Recall	F1
Y	50.89	86.00	63.94
N	54.84	17.00	25.95
Macro F1			44.95

Table 2. Evaluation result on binary-class (BC) classification in Traditional Chinese (CT) data set

	Precision	Recall	F1
Y	50.84	86.17	63.95
N	54.64	16.67	25.54
Macro F1			44.74

Table 3. Evaluation result on multiple-class (MC) classification in Simplified Chinese (CS) data set

	Precision	Recall	F1
F	50.75	68.00	58.12
B	35.03	71.00	46.92
C	33.71	19.67	24.84
I	53.33	2.67	5.08
Macro F1			33.74

Table 4. Evaluation result on multiple-class (MC) classification in Traditional Chinese (CT) data set

	Precision	Recall	F1
F	50.62	68.00	58.04
B	35.35	72.00	47.42
C	33.92	19.33	24.63
I	53.33	2.67	5.08
Macro F1			33.79

5. CONCLUSION

In this paper, we described our system built for the system validation subtask of NTCIR-11 RITE-VAL task. Our system first preprocesses each sentence pair through a series of well-known NLP analysis. We introduced a novel idea of extracting concepts from a sentence. With the preprocessing analysis, we calculate a feature vector for each sentence pair. The feature vectors of labeled sentence pairs are then used for training a SVM model with LibSVM. This model is then used to predict the labels of feature vectors of unlabeled sentence pairs.

6. REFERENCES

[1] Suguru Matsuyoshi and Yusuke Miyao and Tomohide Shibata and Chuan-Jie Lin and Cheng-Wei Shih and Yotaro Watanabe and Teruko Mitamura. Overview of the NTCIR-11 Recognizing Inference in TExt and Validation (RITE-VAL) Task. *Proceedings of the 11th NTCIR Conference*, 2014

[2] Java Chinese Converter. https://code.google.com/p/jcc/ [2014/09/01]

[3] Stanford Word Segmenter. http://nlp.stanford.edu/software/segmenter.shtml [2014/09/01]

[4] Stanford Parser. http://nlp.stanford.edu/software/lex-parser.shtml [2014/09/01]

[5] CKIP Chinese word segmentation system. http://ckipsvr.iis.sinica.edu.tw/ [2014/09/01]

[6] Mei, J., Zhu, Y. Gao, Y. and Yin, H.. (1982). tong2yi4ci2ci2lin2. Shanghai Dictionary Press.

[7] Soujanya Poria, Nir Ofek, Alexander Gelbukh, Amir Hussain and Lior RokachShafqat Mumtaz Virk, Yann-Huei Lee and Lun-Wei Ku. Sentic Demo: A Hybrid Concept Level Aspect Based Sentiment Analysis Toolkit. *Proceedings of the 11th ESWC*, 2014.

[8] C.-C. Chang and C.-J. Lin. LIBSVM : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1--27:27, 2011.