Using MT-Based Metrics for RTE

Alexander Volokh
alvo01@dfki.de

Günter Neumann
neumann@dfki.de

DFKI
Stuhlsatzenhausweg 3
66123 Saarbrücken, Germany

Abstract

We analyse the complexity of the RTE task data and divide the T/H pairs into three different classes, depending on the type of knowledge required to solve the problem. We then propose an approach which is suitable for the easier two classes, which account for two thirds of all pairs. Our assumption is that T and H are translations of the same source sentence. We then use a metric for MT evaluation (Meteor) in order to judge the similarity of both translations. It is clear that in most cases when T entails H, T and H do not have exactly the same meaning. However, we can observe that the similarity is still much higher for positive T/H pairs than for negative pairs. We achieve a result of 46.34 macro-average F1-score for the task. On one hand-side, it shows that our approach has its weaknesses especially because our assumption that T and H contain the same meaning does not always hold, especially if T and H have very different lengths. On the other hand considering the fact that RTE-7 is a difficult class-imbalanced problem (<5% YES, >95% NO) this robust approach achieves a decent result for a large amount of data. It is above the median of this year's results and is comparable with the top results from the previous year.

1 Introduction

RTE-6 has introduced a lot of innovations to the RTE task. Among them are the much larger number of T/H pairs, the natural distribution of the entailment relation among them (<5% YES, >95% NO) and the fact that both T and H have to be considered within the context of the corpus they appear in. This has to led to the fact that both the baselines and the participants' results have dropped significantly in contrast to the first five years of RTE challenges. Top ten systems achieved results between 0.38 and 0.48 F1-score (Bentivogli et al., 2010).

This year we decided to focus on the analysis how good or bad the new results actually are. Furthermore, we wanted to predict an upper bound of the best results that our team could achieve within the time given for the RTE-7 track. Therefore we have decided to study the development data in more details and judge the complexity of the individual T/H pairs according to some complexity classes. After taking a closer look at the data we could divide the instances into following three classes: **A** - syntax, **B** - lexical semantics and **C** - inference. As an example we take one hypothesis H_1 and several different T_i that entail H_1. Depending on the type of knowledge required to infer the entailment relation the Ts can be split into different classes:

H_1: People were forced to leave their pets behind when they evacuated New Orleans.

A:

T_1: Thousands of people were forced to leave their pets behind when they evacuated New Orleans.

B:

T_2: Animal rescue officials have been collecting scores of pets and other animals from the shattered city, while many survivors have told of their distress at having to leave beloved cats and dogs behind in the watery city when they fled.

T_3: Such emotional scenes were repeated perhaps thousands of times along the Gulf Coast last week as pet owners were forced to abandon their animals in the midst of evacuation.

C:

T_4: For Elizabeth Finch, the owner of two dogs named Zorra and Hans Blix, the sight of citizens

forced to choose between their pets and their safety was, like the disaster itself, indicative of broader social rifts.

T_5: The animals are being cared for at a farm north of Louisiana until they can be reunited with their families, many of whom were told they would not be able to bring their pets on evacuation buses and helicopters.

The class **A** is the easiest one - the relevant information is expressed with the same words in both T and H. The maximum that should be done in this case is the analysis of the syntactic structure in order to determine that the structure of T contains the structure of H and thus T entails H, cf. T_1/H_1 pair.

The class **B** is a little bit trickier, e.g. the words used in T_2 and T3 differ from those used in H_1. Thus in order to correctly recognise the entailment relation one has to know about the synonymy of the words *animals* and *pets* or *leave behind* and *abandon* in addition to the syntactic structure.

The class **C** is the most difficult one. For that class of T/H pairs one has to use logic inference and/or world knowledge. For example in order to imply H_1 from T_5 one has to know that *New Orleans* is in *Louisiana.* For the pair H_1/T_4 some deeper logic inference is required in contrast to the rather simple predicate matching of the **A** or **B** classes.

We have manually assigned these classes to about half of all positive instances from the RTE-7 development data. According to our findings around 30% of all positive T/H pairs belonged to **A** and around 35% belonged to **B** and **C** classes each.

Since instances of the class **A** seemed to be an easier case (same words were used in both T and H) it was clear that a meaningful baseline should have an F-score of around 0.3. The second class of instances was more difficult. However, with additional resources for synonymy and/or semantic relatedness information one could hope to achieve a score of at most 0.65. As far as the last class is concerned it seemed to be so diverse and complex that considering our initial situation it was unrealistic for us to come up with a decent solution to it within the scope of this year's task.

Having these considerations in mind we have tried to design a system which would reliably find T/H pairs which express the same information by using the same or synonymous words. The approach which we decided to use was to treat T and H as two different translations of the same source sentence and apply the machine translation evaluation system Meteor (Metric for Evaluation of Translation with Explicit Ordering, cf. Denkowski and Lavie, 2011) to them. This system seemed to be very suitable for our goals, since it is able not only to match the same word forms, but it can also match same stems and even synonyms.

In this paper we describe in details how we analysed the complexity of different T/H pairs and the system we have used to automatically recognise the entailment relation for the RTE-7 challenge data.

2 Entailment Complexity

Recognising textual entailment is a difficult task with a large number of very diverse approaches for dealing with it. At the same time the period of four months that the teams had within the scope of the RTE-7 challenge was too short in order to provide a satisfactory solution for the whole range of problems (at least from scratch). Therefore we have decided to first identify those phenomena which are a) very frequent independently of the topic and b) solvable within several months. This has finally led to the division of the data into the classes **A**, **B** and **C**, as already presented with examples in the introduction.

In many cases it was difficult to tell at a first glance to which of the three classes a T/H pair belongs. Therefore we have designed a procedure for assigning the class label and applied it to 6 of the 10 topics of the RTE-7 development data. This procedure consists of the following three steps:

1. Identify the *relevant predicates* for both T and H.
2. Compare the sets of predicates.
3. Assign the class label.

Here are few examples for the different classes for the T/H pairs used in the introduction.

For the pair T_1/H_1:

1. Relevant predicates for T_1:sbj(people, forced), vc(forced, leave_behind),

obj(leave_behind,pets), sbj(people, evacuated)

Relevant predicates for H_1: sbj(people, forced), vc(forced, leave_behind), obj(leave_behind,pets), sbj(people, evacuated)

2. Comparison: T_1 contains all predicates of H_1
3. Class **A**

For the pair T_2/H_1:

1. Relevant predicates for T_2: sbj(survivors, having), vc(having, leave_behind), obj(leave_behind, cats and dogs), sbj(survivors, fled)

   Relevant predicates for H_1: sbj(people, forced), vc(forced, leave_behind), obj(leave_behind,pets), sbj(people, evacuated)
2. Comparison: T_2 contains all the relevant predicates of H_1. However, a successful match requires synonymy between:

   $\text{survivors} \approx \text{people}$

   $\text{forced} \approx \text{having}$

   $\text{pets} \approx \text{cats\_and\_dogs}$

   $\text{fled} \approx \text{evacuated}$
3. Class **B**

For the pair T_5/H_1:

1. Relevant predicates for T_5: pas_sbj(animals, cared), loc(cared, Louisiana), sbj(families, not_be_able), obj(not_be_able, bring), obj(bring, pets), obj(bring, on_evacuation_buses)

   Relevant predicates for H_1: sbj(people, forced), vc(forced, leave_behind), obj(leave_behind,pets), sbj(people, evacuated), obj(evacuated, New_Orleans)
2. Comparison: the matching of both sets requires inference, e.g.

   $\textit{New Orleans} \approx \textit{Louisiana}$

   $\text{disability to bring pets on evacuation buses} \approx \text{forced to leave behind pets}$
3. Class **C**

It is important to note, that we have not automatised the described procedure and therefore the notions of a *predicate* or *relevance*, as well as the different relation types are informal and do not correspond to any well-defined linguistic theory or formalism. Even though in some cases it was arguable to which category a certain pair belongs, we think that our findings still give a representative impression for the problem difficulty for RTE-7 data.

Our findings have shown that 30% of the pairs belong to the **A** class and 35% belong to the **B** and **C** classes each. Since **C** class is extremely diverse and difficult we thought that it is better to focus on **A** and **B** classes, which amount for two thirds of the data and are easier to solve.

3 Predicting Entailment

For every T/H pair the RTE system is required to predict whether T entails H or not. In order to answer this question we have decided to treat H and T as translations of the same source sentence. Even though it is clear that in most cases when T entails H, T and H do not have exactly the same meaning, but T rather contains more information, the similarity is still much higher in positive T/H pairs than it is in negative ones. Therefore we have used the machine translation evaluation metric Meteor. Meteor is able to align two strings based on exact, stem and synonym matches. This is very suitable for our goals, since T/H pairs of **A** and **B** classes are very similar on the word-level, since only the same or synonymous words are used.

For two strings Meteor returns a score for their similarity. E.g. for T_1 and H_1 Meteor returns the score of 0.63149, whereas for H_1 and some other non-entailing T like "*How can people leave them," wondered Robson, a firefighter from New Mexico who owns a dog and a goldfish.* (AFP_ENG_20050907.0111, sentence 9) the score is only 0.13077.

Additionally, Meteor can log the information about the alignments into a different file. E.g. again for T_1 and H_1 we would get the list of matching fragments in the following form:

0:1	2:1	0	1.0
1:1	3:1	0	1.0
2:1	4:1	0	1.0
3:1	5:1	0	1.0
4:1	6:1	0	1.0
5:1	7:1	0	1.0
6:1	8:1	0	1.0
7:1	9:1	0	1.0
8:1	10:1	0	1.0
9:1	11:1	0	1.0
10:1	12:1	0	1.0
11:1	13:1	0	1.0

12:1 14:1 0 1.0
13:1 15:1 0 1.0

The first column stands for the word index in T, the second column stands for the word index in H, the third column stands for the module which is responsible for the match (0 – exact word match, 1 – stem match, 2 – synonym) and the fourth column stands for the weight of the match (it is configurable that, for instance, exact matches could have higher weights than stem matches, but we do not do that and use Meteor with default settings).

We have used both the original Meteor score, as well as the detailed alignment information in order to define features templates intended to capture the similarity between T and H.

Additionally, we have used numerous feature templates which turned out to be especially useful in our previous system developed during the RTE-6 challenge (Volokh et al., 2010). Among them are the feature templates based on the WordNet similarity measures JCn (Jiang and Conrath, 1997) and Lin (Lin, 1998), as well as the templates based on named entities recognised by Lingpipe (Alias-i, 2008).

4 Feature Model

In this section we describe the features used in our system:

1. We perform a syntactic dependency analysis for both H and T. The dependency representation consists of *triples*: word, the word it depends on and the type of this dependency. We then identify the most *relevant* dependencies: By using the depth of the dependency tree as heuristics: the root of a sentence has the depth 0, root dependents – depth 1, their dependents – 2, etc. We use the depth 2 as our relevance threshold. For all relevant triples in H we look whether there is a triple among relevant triples in T with the same dependency relation. If yes, we calculate their similarity according the JCn and Lin similarity measures. Based on the values we define following two feature templates:

a)
- I) JCn similarity is < 0.5, else
- II) JCn similarity is < 1, else
- III) JCn similarity is < 1.5, else
- IV) JCn similarity is < 2, else
- V) JCn similarity is < 2.5, else
- VI) otherwise

b)
- I) Lin similarity is < 0.2, else
- II) Lin similarity is < 0.4, else
- III) Lin similarity is < 0.6, else
- IV) Lin similarity is < 0.8, else
- V) Lin similarity is < 1, else
- VI) otherwise

2. We annotate all sentences occurring as Ts or Hs with Lingpipe named entity recogniser. We then use the following two feature templates using this annotation:

a)The percentage of the named entities occurring in H also occurring in T:
- I) Less than 20% of named entities occurring in H are present in T, else
- II) Less than 40%, else
- III) Less than 60%,
- IV) Less than 80%
- V) 100%

b)For every type of named entities (PERSON, LOCATION, ORGANIZATION) in H check whether entities of the same type are in T:
- I) There are entities of the same type
- II) There are no entities of the same type

3. We have defined some regular expressions which try to capture the date(day, month, year) of the event described in T or H, e.g. *November 2* or *2005*. In case none of the expressions matches we take the filename of the source document, e.g. LTW_ENG_20050107.0133.xml and extract the date from it (2005, January 7). Then we define a feature template which matches the date of T and H:
- I) if time expressions from H occur in T
- II) otherwise

4. We use Meteor and its detailed alignment information for the following three feature templates:

a) The original Meteor score is used:
- I) Meteor score < 0.1, else
- II) Meteor score < 0.2, else
- III) Meteor score < 0.25, else
- IV) Meteor score < 0.3, else
- V) Meteor score < 0.35, else
- V) Meteor score < 0.4, else
- VI) Meteor score < 0.5, else
- VII) otherwise

b) We use the alignment information to compute the percentage of H contained in T:
- I) less than 20%, else

II) less than 35%, else
III) less than 50%, else
IV) less than 65%, else
V) less than 80%, else
VI) otherwise

c) In order to favour one longer match over several short ones (a matching phrase is better than several individual words all over the sentence), we go through all matching n-grams and compute the following score (1 – length):

$$\frac{l(ngram)}{l(sentence)-l(ngram)+1}$$

We then compute the sum of all scores and use it for our next feature template:

I) the sum is smaller than 0.2, else
II) the sum is smaller than 0.4, else
III) the sum is smaller than 0.6, else
IV) the sum is smaller than 0.8, else
V) otherwise

5. Finally, it is important to consider that we assume that T and H are translations of the same source sentence, despite the fact that in case of entailment T usually contains more information than H. Therefore in some cases our n-gram or Meteor scores can be bad simply because T is much longer than H and contains a lot of irrelevant information. Therefore we define a feature template which compares the length of T and H:

$$c = \frac{l(H)}{l(T)}$$

I) c > 1.5, else
II) c > 1.3, else
III) c > 1.1, else
IV) c > 0.9, else
V) c > 0.7, else
VI) c > 0.5, else
VII) c > 0.3, else
VIII) c > 0.1, else
IX) otherwise

5 Classification

We use the machine learning package called LibLinear(Lin et al., 2008) in order to learn a model for predicting entailment. LibLinear is a collection of linear classification algorithms and the one we use is called L1RLR – L1 regularised logistic regression. This method is very fast and produces very compact models, because due to the regularisation most features get zero weights and do not play any role. We have also tried a more sophisticated classification package by the same group called LivSVM(Lin et al., 2001) but could not reach any improvement in accuracy.

It is very important to keep in mind that the RTE-7 data is a highly imbalanced class problem, since more than 95% of pairs belong to the NO class and only less than 5% are of the YES class. We have tried out many of the typical class-imbalance solutions, including oversampling, undersampling, class cost or decision threshold adjusting.

E.g. for oversampling we have tried to increase the number of YES classes by duplicating the instances of this class in the training data or by introducing new T/H pairs where the hypothesis was paraphrased. A different attempt for undersampling included the elimination of some T/H pairs where T and H were so completely different that the NO answer was absolutely obvious. Thus the overall number of instances decreased, but the proportion of YES classes increased. However, both with oversampling and undersampling we were not able to achieve better results.

The only technique which considerably improved our performance was the manipulation of the decision threshold. Since logistic regression is a probabilistic classification method, one can assign the class label not only if its probability is higher than 50%, but one can arbitrary define this threshold. In our experiments we could obtain best results with the threshold set to 16%.

6 Results

The result of our best submission for the Main task was 43.41 micro-average and 46.34 macro-average F1-score.

We have also submitted some runs for the Novelty Detection task and achieved 80.64 micro-average and 81.03 macro-average F1-score. For these runs exactly the same system, i.e. without absolutely any adaptations to the task, was used.

7 Ablation Tests

We have performed six ablation tests in order to measure how much different feature types presented in section 4 contributed to the overall score.

The following table summarises the results:

Test #	F Measure	Impact	Left out Features
1	42.80	-0.14	WordNet features
2	40.58	2.08	LingPipe features
3	42.71	-0.05	Time regular expressions
4	40.51	2.15	Meteor score feature
5	40.42	2.24	Meteor n-gram feature
6	42.80	-0.14	Length feature

The tests show that Meteor features are the most useful ones in the system. Named entity features have proven beneficial as well. However, other features seem to be not robust enough and were useful only for the development set and did not work for the test data.

8 Conclusions

We have analysed the complexity of RTE-7 data for two reasons. First, we wanted to find out the phenomena which are solvable within the scope of the RTE-7 challenge. Second, we wanted get an objective feeling how good or bad the current results for the task are.

As far the first point is concerned, we have found out that for two thirds of the data one does not require any sophisticated approach and a good word-level analysis can suffice. The rest of the data, however, contains lots of very diverse and difficult phenomena and thus it is unrealistic to come up with a decent solution for it in such a short period of time.

We have obtained an F1-score of 46.34, which is considering our findings far below the targeted 0.65 (upper bound for **A** and **B** classes). On the one hand it is a quite good result, since the data is highly imbalanced and it is very difficult to learn a good model. On the other hand our approach showed many weakness, the main being the inability to deliver reliable results for short Hs and long Ts. The assumption that T and H contain the same information simply did not hold in those cases. Thus at least for these cases a better solution, which considers only the relevant parts of T and H is necessary. Additionally, it is important to investigate how many of the synonyms where actually recognised by the Meteor software and whether there is a necessity to do better at this subtask.

Acknowledgements

The presented work was partially supported by grants from the German Federal Ministry of Economics and Technology (BMWi) to the DFKI THESEUS project (FKZ: 01MQ07016).

References

Alias-i. 2008. **LingPipe 4.0.0**. *http://alias-i.com/lingpipe*

L. Bentivogli, P. Clark, I. Dagan, H.T. Dang, D. Giampiccolo. ***Overview of the TAC 2010 Summarization Track.*** *Proceedings of the Text Analysis Conference (TAC 2010). Gaithersburg, Maryland, USA.*

Michael Denkowski and Alon Lavie. **"Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems"**, *Proceedings of the EMNLP 2011 Workshop on Statistical Machine Translation, 2011*

Chang, C.-C. and C.-J. Lin (2001). **LIBSVM: A Library for Support Vector Machines**. *Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm*

Alexander Volokh, Günter Neumann, Bogdan Sacaleanu. ***Combining Deterministic*** **Dependency** ***Parsing*** **and Linear Classifica- tion for Robust RTE**. *Proceedings of the Text Analysis Conference (TAC 2010). Gaithersburg, Maryland, USA.*