UvT: Memory-based pairwise ranking of paraphrasing verbs

Sander Wubben
Tilburg centre for Cognition and Communication
Tilburg University
The Netherlands
s.wubben@uvt.nl

Abstract

In this paper we describe Mephisto, our system for Task 9 of the SemEval-2 workshop. Our approach to this task is to develop a machine learning classifier which determines for each verb pair describing a noun compound which verb should be ranked higher. These classifications are then combined into one ranking. Our classifier uses features from the Google N-gram Corpus, WordNet and the provided training data.

1 Introduction

We interpret the task of ranking a set of given paraphrasing verbs as described by Butnariu et al (2010) as a competition between these verbs. Each verb competes with every other verb in the set and receives a positive score if it is more likely to describe the given noun compound (NC) than the other verb and a negative score if it is less likely to describe the NC. In line with this approach we regard the task as a classification problem where for each comparison our classification algorithm picks the paraphrasing verb that is more likely to describe the NC. This brings the classification problem down to three classes: higher, equal or lower. Sometimes the paraphrasing verbs are accompanied by a preposition. In this paper we will simply refer to all verbs and verb-prepositions as verbs.

The distribution of the verbs in the training data provides us already with valuable information. We incorporate basic features describing this distribution to train our classifier. We also need additional semantic features that provide us with insight into the relation between the NC and the verb, therefore we use features constructed from WordNet and the Google N-gram Corpus to train our Memory-based paraphrase interpretation scoring tool (Mephisto).

2 System Description

The system consists of three components: the feature extraction component, the classification component and the ranking component. We will describe all three components.

2.1 Feature Extraction

For each verb describing an NC we try to extract those features that describe the probability that this verb is a good interpretation of the NC. We assume that given a NC N_1N_2 and a verb V, the NC interpretation should be N_2VN_1. The phrase "Butter made from peanuts" adequately describes peanut butter.

The training data provides us with a total of 17,727 instances of NC verb pairs scored by human judges. This can be broken down into 4,360 unique verb phrases describing 250 NCs. This distribution already gives us a good clue when we are generating new rankings. The following are the features we used:

Weighted mean in training data For each NC that has to be ranked we find the most similar NC in the training data by measuring the overlap in verb phrases between the two NCs. We do this by calculating the Jaccard coefficient over the sets of verbs associated with the NCs. We adapt the highest ranking NC as most similar to our candidate NC (the NC with most matching verbs). For each verb V we then calculate the score as follows:

$$Score = J * S_{sim} + (1 - J) * M$$

where J is the Jaccard score, S_{sim} is the assigned score of the verb in the most similar set and M is the mean score for the verb in the training data.

Rank in training data For this feature we directly compare the two verbs V_1 and V_2. We just

Proceedings of the 5th International Workshop on Semantic Evaluation, ACL 2010, pages 260–263,
Uppsala, Sweden, 15-16 July 2010.

feature	values	info gain	gain ratio
verb1	4,093	0.24	0.02
verb2	4,093	0.24	0.02
verb1-verb2	768,543	1.06	0.06
verb1-verb2-LCS	986,031	1.29	0.07
n-gram score1	7	0.07	0.02
n-gram score2	7	0.01	0.08
weighted mean	7	0.29	0.12
rank	3	0.68	0.43

Table 1: Features used in our system

count the number of times that V_1 is ranked higher than V_2 and vice versa for every NC where both verbs occur. We end up with a positive, equal or negative class.

WordNet Least Common Subsumer In order to distinguish between different kinds of NCs we use WordNet (Fellbaum, 1998) to determine the kind of relation between the nouns. This idea is supported by work by Levi (1978), Warren (1978) and Nastase & Szpakowicz (2003). Our intuition is that the ranking of verb phrases is very dependent on this relation between the nouns. To determine this we use the WordNet::QueryData (Rennie, 2000) module. In the WordNet graph we look for the Least Common Subsumer (LCS) of the two nouns. The LCS is the lowest parent node of both nouns. We combine the LCS with both verb phrases into one feature.

Google N-gram features We use the Google N-gram corpus to count co-occurence frequencies of certain n-grams. An NC occurring often together with a certain verb should indicate that that verb is a good paraphrase for the NC. Using web text for various NLP-tasks has been proven to be useful (Lapata and Keller, 2005), also for NC interpretation (Nakov and Hearst, 2005). Because of data sparseness and the unlikelihood of finding a perfect match for a certain n-gram, we adopt different strategies for constructing features. First of all, we try to relax the matching conditions by applying certain regular expression. Given the NC "abortion problem" and the paraphrasing verb "be related to" , it seems unlikely you will ever encounter the n-gram "problem be related to abortion", yet in the training data "be related to" is the number three verb for "abortion problem". Therefore, we first apply some simple inflection. Instead of "be" we match on "is/are/being". and we do a comparable inflection for other verbs transforming

	+up+	-dwn-	=eq=
+up+	23,494	7,099	8,912
-dwn-	7,168	23,425	8,912
=eq=	22,118	22,084	22,408

Table 2: Confusion matrix of the classes, with horizontally the output classes and vertically the target classes

a verb such as "involve" into "involves/involving". Additionally we also match on singular and plural nouns. We then use two different techniques to find the n-gram frequencies:

$$N-gram_1 = \frac{f(N_2V) + f(VN_1)}{f(V)}$$

$$N-gram_2 = \frac{f(N_2VN_1)}{f(V)}$$

where f stands for the occurrences of the given sequences of nouns and verb. We do not divide by noun occurrences because they are constant for every pair of verbs we compare.

Pairwise comparison of features For each verb pair in an NC set we compare all numeric features and assign one of the following symbols to characterize the relation of the two verbs:

+++: V_1 score is more than 10 times V_2 score
++: V_1 score is between 2 and 10 times V_2 score
+: V_1 score is between 1 and 2 times verb2 score
=: scores are equal
-: V_2 score is between 1 and 2 times V_1 score
- -: V_2 score is between 2 and 10 times V_1 score
- - -: V_2 score is more than 10 times V_1 score

An overview of the features is displayed in Table 1.

2.2 Classification

Our system makes use of Memory-Based Learning (MBL) for classification. MBL stores feature representations of training instances in memory without abstraction and classifies unseen instances by matching their feature representation to all instances in memory, finding the most similar instances. The class of these most similar instances is then copied to the new instance The learning algorithm our system uses is the IB1 classier as implemented in TiMBL (version 6.1.5). IB1 is a supervised decision-tree-based implementation of

Settings	TiMBL F-score	Spearman ρ	Pearson r	KullbackLeibler div.
k=3 all features	0.48	**0.50**	0.44	1.91
k=3 no external features	0.53	0.48	0.41	2.05
k=11 all features	0.51	0.50	0.42	1.97
k=11 no external features	0.20	-	-	-

Table 3: Results for different settings on the development set

the k-nearest neighbor algorithm for learning classification tasks (Aha et al., 1991). The TiMBL parameters we used in the Mephisto system for the IB1 classifier are the overlap metric, weighting using GainRatio, and k=3, taking into account the instances on the 3 most similar positions to extrapolate the class of the instance. More information about these settings can be found in the TiMBL reference guide (Daelemans et al., 2009). We train our classifier on the provided training data to classify instances into one of three classes; **+up+** if V_1 ranks higher than V_2 , **=eq=** if both verbs rank equally and **-dwn-** if V_1 ranks lower than V_2.

2.3 Ranking

The final step is to combine all the classification into one score per verb. This is done in a very straight forward way: a verb receives one point every time it is classified as +up+. This results in scores for each verb paraphrasing an NC. We then perform a simple post processing step: we reassign classes to each verb based on the final scores they have received and recalculate their scores. We repeat this process until the scores converge.

3 Results

For development the original training set was divided in a development training set of 15,966 lines and a development test set of 1,761 lines, which contains 23 NCs. The distribution and ranking features were calculated using only the development training set. Because we compare for each NC every verb to every other verb the TiMBL training instance-base contains 1,253,872 lines, and the development test set 145,620. The results for different settings are in Table 3. Although the TiMBL F-score (macro-averaged) of using all features is actually lower than using only semantic features at k=3, the final correlations are in favor of using all features. There does not seem to be an improvement when extrapolating from 11 neighbouring instances in the instance-base over 3. In fact, when using no external features and k=11, the classifier overgeneralizes and classifies every instance as =eq= and consequently does not provide a ranking

System	Spearman ρ	Pearson r	Cosine
UvT-MEPHISTO	0.450	0.411	0.635
UCD-PN	0.441	0.361	0.669
UCD-GOGGLE-III	0.432	0.395	0.652
UCD-GOGGLE-II	0.418	0.375	0.660
UCD-GOGGLE-I	0.380	0.252	0.629
UCAM	0.267	0.219	0.374
NC-INTERP	0.186	0.070	0.466
Baseline	0.425	0.344	0.524

Table 4: Final results for SemEval-2 Task 9

at all. Additionally, classifying with k=11 takes considerably longer than with k=3. The settings we use for our final system are k=3 and we use all features. Table 2 displays a confusion matrix of the classification on the development test set. Not surprisingly the classifier is very bad at recognizing the =eq= class. These mistakes are not as bad as miss-classifying a +up+ instance as -dwn- and vice versa, and fortunately these mistakes happen less often.

The official test set contains 32,830 instances, almost twice as many as the training set. This breaks down into 2,837,226 cases to classify. In Table 4 are the final results of the task with all participating systems and their macro-averaged Spearman, Pearson and Cosine correlation. Also shown is the baseline, which involves scoring a given verb paraphrase by its frequency in the training set. The final results are quite a bit lower than the results on the development set. This could be coincidence (the final test set is about twenty times larger than our development test set), but it could also be due to overfitting on the development set. The ten best and worst scoring compounds are shown in Table 5 with their Least Common Subsumer as taken from WordNet. The best-scoring NC "jute products" achieves a Spearman ρ of 0.75 while the worst-scoring compound, "electron microscope" only achieves 0.12.

4 Conclusion

We have shown that a Memory-based pairwise approach to ranking with features taken from WordNet and the Google N-gram corpus achieves

Best scoring NCs	LCS	Spearman ρ
jute products	physical entity	0.75
ceramics products	artifact	0.75
steel frame	physical entity	0.74
cattle population	entity	0.74
metal body	physical entity	0.74
winter blooming	entity	0.73
warbler family	entity	0.72
wool scarf	artifact	0.71
fiber optics	physical entity	0.70
petroleum products	physical entity	0.70
Worst scoring NCs	**LCS**	**Spearman ρ**
electron microscope	whole	0.12
light bulb	physical entity	0.15
yesterday evening	measure	0.16
student loan	entity	0.16
theater orchestra	entity	0.17
sunday restrictions	abstraction	0.20
yesterday afternoon	measure	0.20
relations agency	abstraction	0.21
crime novelist	entity	0.21
office buildings	structure	0.21

Table 5: Best and worst scoring noun compounds with their Least Common Subsumer and Spearman ρ correlation

good results on the task of ranking verbs paraphrasing noun compounds. We outperform the strong baseline and also systems using an unsupervised approach. If we analyse our results we see that our system scores particularly well on noun compounds describing materials: in Table 5 we see that all top ten compounds are either "artifacts", "physical entities" or "entities" according to WordNet and the relation is quite direct: generally a *made of* relation seems appropriate. If we look at the bottom ten on the other hand, we see relations such as "abstraction" and "measure": these are harder to qualify. Also, an "electron microscope" will generally not be perceived as a microscope made of electrons. We can conclude that for NCs where the relation between the nouns is more obscure the verbs are harder to rank.

If we look at the Information Gain Ratio, of all features the rank difference of the verbs in the training data seems to be the strongest feature, and of the external features the frequency difference of the entire phrase containing the NC and the verb. A lot more investigations could be made into the viability of using large n-gram collections such as the Google N-gram corpus for paraphrase tasks.

It might also be interesting to explore a somewhat more challenging variant of this task by not providing the verbs to be ranked a priori. This would probably be more interesting for real world applications because often the task is not only ranking but finding the verbs in the first place. Our system should be able to handle this task with minor modifications: we simply regards all verbs in the training-data candidates to be ranked. Then, a pre-filtering step should take place to weed out irrelevant verbs based on an indicator such as the LCS of the nouns. In addition a threshold could be implemented to only accept a (further) limited set of verbs in the final ranking.

References

David W. Aha, Dennis Kibler, and Marc K. Albert. 1991. Instance-based learning algorithms. *Mach. Learn.*

Cristina Butnariu, Su Nam Kim, Preslav Nakov, Diarmuid Ó Séaghdha, Stan Szpakowicz, and Tony Veale. 2010. Semeval-2 task 9: The interpretation of noun compounds using paraphrasing verbs and prepositions. In *Proceedings of the 5th SIGLEX Workshop on Semantic Evaluation*.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2009. Timbl: Tilburg memory-based learner - version 6.2 - reference guide.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*.

Mirella Lapata and Frank Keller. 2005. Web-based models for natural language processing. *ACM Trans. Speech Lang. Process.*

Judith N. Levi. 1978. *The Syntax and Semantics of Complex Nominals*.

Preslav Nakov and Marti Hearst. 2005. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *Proceedings of the 9th Conference on Computational Natural Language Learning*.

Vivi Nastase and Stan Szpakowicz. 2003. Exploring noun-modifier semantic relations. In *Proceedings of the 5th International Workshop on Computational Semantics*.

Jason Rennie. 2000. Wordnet::querydata: a perl module for accessing the wordnet database.

Beatrice Warren. 1978. *Semantic Patterns of Noun-Noun Compounds*.