The Meaning Factory: Formal Semantics for Recognizing Textual Entailment and Determining Semantic Similarity

Johannes Bjerva
Univ. of Groningen
j.bjerva@rug.nl

Johan Bos
Univ. of Groningen
johan.bos@rug.nl

Rob van der Goot
Univ. of Groningen
r.van.der.goot@rug.nl

Malvina Nissim
Univ. of Bologna
malvina.nissim@unibo.it

Abstract

Shared Task 1 of SemEval-2014 comprised two subtasks on the same dataset of sentence pairs: recognizing textual entailment and determining textual similarity. We used an existing system based on formal semantics and logical inference to participate in the first subtask, reaching an accuracy of 82%, ranking in the top 5 of more than twenty participating systems. For determining semantic similarity we took a supervised approach using a variety of features, the majority of which was produced by our system for recognizing textual entailment. In this subtask our system achieved a mean squared error of 0.322, the best of all participating systems.

1 Introduction

The recent popularity of employing distributional approaches to semantic interpretation has also lead to interesting questions about the relationship between classic formal semantics (including its computational adaptations) and statistical semantics. A promising way to provide insight into these questions was brought forward as Shared Task 1 in the SemEval-2014 campaign for semantic evaluation (Marelli et al., 2014). In this task, a system is given a set of sentence pairs, and has to predict for each pair whether the sentences are somehow related in meaning. Interestingly, this is done using two different metrics: the first stemming from the formal tradition (contradiction, entailed, neutral), and the second in a distributional fashion (a similarity score between 1 and 5). We participated in this shared task with a system rooted in formal semantics. In particular, we were interested in finding out whether paraphrasing techniques could increase the accuracy of our system, whether meaning representations used for textual entailment are useful for predicting semantic similarity, and conversely, whether similarity features could be used to boost accuracy of recognizing textual entailment. In this paper we outline our method and present the results for both the textual entailment and the semantic similarity task.[1]

2 Recognizing Textual Entailment

2.1 Overview

The core of our system for recognizing textual entailment works as follows: (i) produce a formal semantic representation for each sentence for a given sentence pair; (ii) translate these semantic representations into first-order logic; (iii) use off-the-shelf theorem provers and model builders to check whether the first sentence entails the second, or whether the sentences are contradictory. This is essentially an improved version of the framework introduced by Bos & Markert (2006).

To generate background knowledge that could assist in finding a proof we used the lexical database WordNet (Fellbaum, 1998). We also used a large database of paraphrases (Ganitkevitch et al., 2013) to alter the second sentence in case no proof was found at the first attempt, inspired by Bosma & Callison-Burch (2006). The core system reached high precision on entailment and contradiction. To increase recall, we used a classifier trained on the output from our similarity task system (see Section 3) to reclassify the "neutrals" into possible entailments.

2.2 Technicalities

The semantic parser that we used is Boxer (Bos, 2008). It is the last component in the pipeline of the C&C tools (Curran et al., 2007), comprising a tokenizer, POS-tagger, lemmatizer (Minnen et

[1]To reproduce these results in a linux environment (with SWI Prolog) one needs to install the C&C tools (this includes Boxer and the RTE system), the Vampire theorem prover, the two model builders Paradox and Mace-2, and the PPDB-1.0 XL database. Detailed instructions can be found in the `src/scripts/boxer/sick/README` folder of the C&C tools.

al., 2001), and a robust parser for CCG (Steedman, 2001). Boxer produces semantic representations based on Discourse Representation Theory (Kamp and Reyle, 1993). We used the standard translation from Discourse Representation Structures to first-order logic, rather than the one based on modal first-order logic (Bos, 2004), since the shared task data did not contain any sentences with propositional argument verbs.

After conversion to first-order logic, we checked with the theorem prover Vampire (Riazanov and Voronkov, 2002) whether a proof could be found for the first sentence entailing the second, and whether a contradiction could be detected for the conjunction of both sentences translated into first-order logic. If neither a proof nor a contradiction could be found within 30 seconds, we used the model builder Paradox (Claessen and Sörensson, 2003) to produce a model of the two sentences separately, and one of the two sentences together. However, even though Paradox is an efficient piece of software, it does not always return minimal models with respect to the extensions of the non-logical symbols. Therefore, in a second step, we asked the model builder Mace-2 (McCune, 1998) to construct a minimal model for the domain size established by Paradox. These models are used as features in the similarity task (Section 3).

Background knowledge is important to increase recall of the theorem prover, but hard to acquire automatically (Bos, 2013). Besides translating hypernym relations of WordNet to first-order logic axioms, we also reasoned that it would be beneficial to have a way of dealing with multi-word expressions. But instead of translating paraphrases into axioms, we used them to rephrase the input sentence in case no proof or contradiction was found for the original sentence pair. Given a paraphrase SRC$\mapsto$TGT, we rephrased the first sentence of a pair only if SRC matches with up to four words, no words of TGT were already in the first sentence, and every word of TGT appeared in the second sentence. The paraphrases themselves were taken from PPDB-1.0 (Ganitkevitch et al., 2013). In the training phrase we found that the XL version (comprising o2m, m2o, phrasal, lexical) gave the best results (using a larger version caused a strong decrease in precision, while smaller versions lead to a decrease in recall).

We trained a separate classifier in order to reclassify items judged by our RTE system as being neutral. This classifier uses a single feature, namely the relatedness score for each sentence pair. As training material, we used the gold relatedness scores from the training and trial sets. For classification of the test set, we used the relatedness scores obtained from our Semantic Similarity system (see Section 3). The classifier is a Support Vector Machine classifier, in the implementation provided by *Scikit-Learn* (Pedregosa et al., 2011), based on the commonly used implementation *LIBSVM* (Chang and Lin, 2011). We used the implementation's standard parameters.

2.3 Results

We submitted two runs. The first (primary) run was produced by a configuration that included reclassifying the 'neutrals'. The second run is without the reclassification of the neutrals. After submission we ran a system that did not use the paraphrasing technique in order to measure what influence the PPDB had on our performance. The results are summarized in Table 1. In the training phase we got the best results for the configuration using the PPDB and reclassication, which was submitted as our primary run.

Table 1: Results on the entailment task for various system configurations.

System Configuration	**Accuracy**
most frequent class baseline	56.7
−PPDB, −reclassification	77.6
+PPDB, −reclassification	79.6
+PPDB, +reclassification	81.6

In sum, our system for recognizing entailment performed well reaching 82% accuracy and by far outperforming the most-frequent class baseline (Table 1). We show some selected examples illustrating the strengths of our system below.

Example 1627 (ENTAILMENT)
A man is mixing a few ingredients in a bowl
Some ingredients are being mixed in a bowl by a person

Example 2709 (CONTRADICTION)
There is no person boiling noodles
A woman is boiling noodles in water

Example 9051 (ENTAILMENT)
A pair of kids are sticking out blue and green colored tongues
Two kids are sticking out blue and green colored tongues

A proof for entailment is found for Ex. 1627, because for passive sentences Boxer produces a meaning representation equivalent to their active variants. A contradiction is detected for Ex. 2709 because of the way negation is handled by Boxer. Both examples trigger background knowledge from WordNet hyperonyms (man $\rightarrow$ person; woman $\rightarrow$ person) that is used in the proofs.[2] Ex. 9051 shows how paraphrasing helps, here "a pair of" $\mapsto$ "two".

3 Determining Semantic Similarity

3.1 Overview

The Semantic Similarity system follows a supervised approach to solving the regression problem of determining the similarity between each given sentence pair. The system uses a variety of features, ranging from simpler ones such as word overlap, to more complex ones in the form of deep semantic features and features derived from a compositional distributional semantic model. The majority of these features are derived from the models from our RTE system (see Section 2).

3.2 Technicalities

3.2.1 Regressor

The regressor used is a Random Forest Regressor in the implementation provided by *Scikit-Learn* (Pedregosa et al., 2011). Random forests are robust with respect to noise and do not overfit easily (Breiman, 2001). These two factors make them a highly suitable choice for our approach, since we are dealing with a relatively large number of weak features, i.e., features which may be seen as individually containing a rather small amount of information for the problem at hand.

Our parameter settings for the regressor is follows. We used a total of 1000 trees, with a maximum tree depth of 20. At each node in a tree the regressor looked at maximum 3 features in order to decide on the split. The quality of each such split is determined using mean squared error as measure. These parameter values were optimised when training on the training set, with regards to performance on the trial set.

3.2.2 Feature overview

We used a total of 32 features for our regressor. Due to space constraints, we have sub-divided our features into groups by the model/method involved. For all features we compared the outcome of the original sentence pair with the outcome of the paraphrased sentence pairs (see Section 2.2)[3]. If the paraphrased sentence pair yielded a higher feature overlap score than the original sentence pair, we utilized the former. In other words, we assume that the sentence pair generated with paraphrases is a good representation of the original pair, and that similarities found here are an improvement on the original score.

Logical model We used the logical models created by Paradox and Mace for the two sentences separately, as well as a combined model (see Section 2.2). The features extracted from this model are the proportion of overlap between the instances in the domain, and the proportion of overlap between the relations in the model.

Noun/verb overlap We first extracted and lemmatised all nouns and verbs from the sentence pairs. With these lemmas we calculated two new separate features, the overlap of the noun lemmas and the overlap of the verb lemmas.

Discourse Representation Structure (DRS) The two most interesting pieces of information which easily can be extracted from the DRS models are the agents and patients. We first extracted the agents for both sentences in a sentence pair, and then computed the overlap between the two lists of agents. Secondly, since all sentences in the corpus have exactly one patient, we extracted the patient of each sentence and used this overlap as a binary feature.

Wordnet novelty We build one tree containing all WordNet concepts included in the first sentence, and one containing all WordNet concepts of both sentences together. The difference in size between these two trees is used as a feature.

RTE The result from our RTE system (entailment, neutral or contradiction) is used as a feature.

Compositional Distributional Semantic Model Our CDSM feature is based on word vectors derived using a Skip-Gram model (Mikolov et al., 2013a; Mikolov et al., 2013b). We used the publicly available *word2vec*[4] tool to calculate these vectors. We trained the tool on a data set consisting of the first billion characters of Wikipedia[5] and the English part of the French-English 10^9 corpus used in the wmt11 translation task[6]. The Wikipedia section of the data was pre-processed using a script[7] which made the text lower case, removed tables etc. The second section of the data was also converted to lower case prior to training.

We trained the vectors using the following parameter settings. Vector dimensionality was set

[2]In the training data around 20% of the proofs for entailment were established with the help of WordNet, but only 4% for detecting contradictions.

[3]In addition to the PPDB we added handling of negations, by removing some negations {not, n't} and substituting others {no:a, none:some, nobody:somebody}.

[4]code.google.com/p/word2vec/

[5]mattmahoney.net/dc/enwik9.zip

[6]statmt.org/wmt11/translation-task.html#download

[7]mattmahoney.net/dc/textdata.html

Table 2: Pearson correlation and MSE obtained on the test set for each feature group in isolation.

Feature group	p [−PPDB]	p [+PPDB]	MSE [−PPDB]	MSE [+PPDB]
Logical model	0.649	0.737	0.590	0.476
Noun/verb overlap	0.647	0.676	0.592	0.553
DRS	0.634	0.667	0.610	0.569
Wordnet novelty	0.652	0.651	0.590	0.591
RTE	0.621	0.620	0.626	0.627
CDSM	0.608	0.609	0.681	0.679
IDs	0.493	0.493	0.807	0.807
Synset	0.414	0.417	0.891	0.889
Word overlap	0.271	0.340	0.944	0.902
Sentence length	0.227	0.228	0.971	0.971
All with IDs	0.836	0.842	0.308	0.297
All without IDs	0.819	**0.827**	0.336	**0.322**

to 1600 with a context window of 10 words. The skip-gram model with hierarchical softmax, and a negative sampling of $1e$-3 was used.

To arrive at the feature used for our regressor, we first calculated the element-wise sum of the vectors of each word in the given sentences. We then calculated the cosine distance between the sentences in the sentence pair.

IDs One surprisingly helpful feature was each sentence pair's ID in the corpus.[8] Since this feature clearly is not representative of what one would have access to in a real-world scenario, it was not included in the primary run.

Synset Overlap We built one set for each sentence pair consisting of each possible lemma form of all possible noun synsets for each word. The proportion of overlap between the two resulting sets was then used as a feature. Given cases where relatively synonymous words are used (e.g. *kid* and *child*), these will often belong to the same synset, thus resulting in a high overlap score.

Synset Distance We first generated each possible word pair consisting of one word from each sentence. Using these pairings, we calculated the maximum *path similarity* between the noun synsets available for these words. This calculation is restricted so that each word in the first sentence in each pair is only used once.

Word overlap Our word overlap feature was calculated by first creating one set per sentence, containing each word occurring in that sentence. The four most common words in the corpus were used as a stop list, and removed from each set. The proportion of overlap between the two sets was then used as our word overlap feature.

Sentence Lengths The difference in length between the sentence pairs proved to be a somewhat useful feature. Although mildly useful for this particular data set, we do not expect this to be a particularly helpful feature in real world applications.

3.3 Results

We trained our system on 5000 sentence pairs, and evaluated it on 4927 sentence pairs. Table 2 contains our scores for the evaluation, broken up per feature group. Our relatedness system yielded the highest scores compared to all other systems in this shared task, as measured by MSE and Spearman correlation scores. Although our system performed slightly worse as measured by Pearson correlation, there is no significant difference to the scores obtained by the two higher ranked systems.

4 Conclusion

Our work shows that paraphrasing techniques can be used to improve the results of a textual entailment system. Additionally, the scores from our semantic similarity measure could be used to improve the scores of the textual entailment system. Our work also shows that deep semantic features can be used to predict semantic relatedness.

Acknowledgements

We thank Chris Callison-Burch, Juri Ganitkevitch and Ellie Pavlick for getting the most out of PPDB. We also thank our colleagues Valerio Basile, Harm Brouwer, Kilian Evang and Noortje Venhuizen for valuable comments and feedback.

[8] We discovered that the ordering of the entire data set was informative for the prediction of sentence relatedness. We have illustrated this by using the ordering of the sentences (i.e. the sentence IDs) as a feature in our model, and thereby obtaining better results. Relying on such a non-natural ordering of the sentences would be methodologically flawed, and therefore this feature was not used in our primary run.

References

Johan Bos and Katja Markert. 2006. Recognising textual entailment with robust logical inference. In Joaquin Quinonero-Candela, Ido Dagan, Bernardo Magnini, and Florence d'Alché Buc, editors, *Machine Learning Challenges, MLCW 2005*, volume 3944 of *LNAI*, pages 404–426.

Johan Bos. 2004. Computational Semantics in Discourse: Underspecification, Resolution, and Inference. *Journal of Logic, Language and Information*, 13(2):139–157.

Johan Bos. 2008. Wide-Coverage Semantic Analysis with Boxer. In J. Bos and R. Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, volume 1 of *Research in Computational Semantics*, pages 277–286. College Publications.

Johan Bos. 2013. Is there a place for logic in recognizing textual entailment? *Linguistic Issues in Language Technology*, 9(3):1–18.

Wauter Bosma and Chris Callison-Burch. 2006. Paraphrase substitution for recognizing textual entailment. In *Proceedings of CLEF*.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

K. Claessen and N. Sörensson. 2003. New techniques that improve mace-style model finding. In P. Baumgartner and C. Fermüller, editors, *Model Computation – Principles, Algorithms, Applications (Cade-19 Workshop)*, pages 11–27, Miami, Florida, USA.

James Curran, Stephen Clark, and Johan Bos. 2007. Linguistically Motivated Large-Scale NLP with C&C and Boxer. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 33–36, Prague, Czech Republic.

Christiane Fellbaum, editor. 1998. *WordNet. An Electronic Lexical Database*. The MIT Press.

Juri Ganitkevitch, Benjamin VanDurme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2013)*, Atlanta, Georgia, June. Association for Computational Linguistics.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. Kluwer, Dordrecht.

M. Marelli, L. Bentivogli, M. Baroni, R. Bernardi, S. Menini, and R. Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of SemEval 2014: International Workshop on Semantic Evaluation.*

W. McCune. 1998. Automatic Proofs and Counterexamples for Some Ortholattice Identities. *Information Processing Letters*, 65(6):285–291.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of english. *Journal of Natural Language Engineering*, 7(3):207–223.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

A. Riazanov and A. Voronkov. 2002. The Design and Implementation of Vampire. *AI Communications*, 15(2–3):91–110.

Mark Steedman. 2001. *The Syntactic Process*. The MIT Press.