UCAM-CORE: Incorporating structured distributional similarity into STS

Tamara Polajnar Laura Rimell Douwe Kiela
Computer Laboratory
University of Cambridge
Cambridge CB3 0FD, UK
{tamara.polajnar,laura.rimell,douwe.kiela}@cl.cam.ac.uk

Abstract

This paper describes methods that were submitted as part of the *SEM shared task on Semantic Textual Similarity. Multiple kernels provide different views of syntactic structure, from both tree and dependency parses. The kernels are then combined with simple lexical features using Gaussian process regression, which is trained on different subsets of training data for each run. We found that the simplest combination has the highest consistency across the different data sets, while introduction of more training data and models requires training and test data with matching qualities.

1 Introduction

The Semantic Textual Similarity (STS) shared task consists of several data sets of paired passages of text. The aim is to predict the similarity that human annotators have assigned to these aligned pairs. Text length and grammatical quality vary between the data sets, so our submissions to the task aimed to investigate whether models that incorporate syntactic structure in similarity calculation can be consistently applied to diverse and noisy data.

We model the problem as a combination of kernels (Shawe-Taylor and Cristianini, 2004), each of which calculates similarity based on a different view of the text. State-of-the-art results on text classification have been achieved with kernel-based classification algorithms, such as the support vector machine (SVM) (Joachims, 1998), and the methods here can be adapted for use in multiple kernel classification, as in Polajnar et al. (2011). The kernels are combined using Gaussian process regression (GPR) (Rasmussen and Williams, 2006). It is important to note that the combination strategy described here is only a different way of viewing the regression-combined mixture of similarity measures approach that is already popular in STS systems, including several that participated in previous SemEval tasks (Croce et al., 2012; Bär et al., 2012). Likewise, others, such as Croce et al. (2012), have used tree and dependency parse information as part of their systems; however, we use a tree kernel approach based on a novel encoding method introduced by Zanzotto et al. (2011) and from there derive two dependency-based methods.

In the rest of this paper we will describe our system, which consists of distributional similarity (Section 2.1), several kernel measures (Section 2.2), and a combination method (Section 2.3). This will be followed by the description of our three submissions (Section 3), and a discussion of the results (Section 4).

2 Methods

At the core of all the kernel methods is either surface, distributional, or syntactic similarity between sentence constituents. The methods themselves encode sentences into vectors or sets of vectors, while the similarity between any two vectors is calculated using cosine.

2.1 Distributional Similarity

Target words are the non-stopwords that occur within our training and test data. The two distributional methods we use here both represent target

words as vectors that encode word occurrence within a set of contexts. The first method is a variation on BEAGLE (Jones and Mewhort, 2007), which considers contexts to be words that surround targets. The second method is based on ESA (Gabrilovich and Markovitch, 2007), which considers contexts to be Wikipedia documents that contain target words.

To gather the distributional data with both of these approaches we used 316,305 documents from the September 2012 snapshot of Wikipedia. The training corpus for BEAGLE is generated by pooling the top 20 documents retrieved by querying the Wikipedia snapshot index for each target word in the training and test data sets.

2.1.1 BEAGLE

Random indexing (Kaski, 1998) is a technique for dimensionality reduction where pseudo-orthogonal bases are generated by randomly sampling a distribution. BEAGLE is a model where random indexing is used to represent word co-occurrence vectors in a distributional model.

Each context word is represented as a D-dimensional vector of normally distributed random values drawn from the Gaussian distribution

$$\mathcal{N}(0, \sigma^2), \text{ where } \sigma = \frac{1}{\sqrt{D}} \text{ and } D = 4096 \quad (1)$$

A target word is represented as the sum of the vectors of all the context words that occur within a certain context window around the target word. In BEAGLE this window is considered to be the sentence in which the target word occurs; however, to avoid segmenting the entire corpus, we assume the window to include 5 words to either side of the target. This method has the advantage of keeping the dimensionality of the context space constant even if more context words are added, but we limit the context words to the top 10,000 most frequent non-stopwords in the corpus.

2.1.2 ESA

ESA represents a target word as a weighted ranked list of the top N documents that contain the word, retrieved from a high quality collection. We used the BM25F (Robertson et al., 2004) weighting function and the top $N = 700$ documents. These parameters were chosen by testing on the WordSim353 dataset.[1] The list of retrieved documents can be represented as a very sparse vector whose dimensions match the number of documents in the collection, or in a more computationally efficient manner as a hash map linking document identifiers to the retrieval weights. Similarity between lists was calculated using the cosine measure augmented to work on the hash map data type.

2.2 Kernel Measures

In our experiments we use six basic kernel types, which are described below. Effectively we have eight kernels, because we also use the tree and dependency kernels with and without distributional information. Each kernel is a function which is passed a pair of short texts, which it then encodes into a specific format and compares using a defined similarity function. LK uses the regular cosine similarity function, but LEK, TK, DK, MDK, DGK use the following cosine similarity redefined for sets of vectors. If the texts are represented as sets of vectors X and Y, the set similarity kernel function is:

$$\kappa_{set}(X, Y) = \sum_i \sum_j \cos(\vec{x}_i, \vec{y}_j) \quad (2)$$

and normalisation is accomplished in the standard way for kernels by:

$$\kappa_{set-n}(X, Y) = \frac{\kappa_{set}(X, Y)}{\sqrt{(\kappa_{set}(X, X)\kappa_{set}(Y, Y))}} \quad (3)$$

LK - The **lexical kernel** calculates the overlap between the tokens that occur in each of the paired texts, where the tokens consist of Porter stemmed (Porter, 1980) non-stopwords. Each text is represented as a frequency vector of tokens that occur within it and the similarity between the pair is calculated using cosine.

LEK - The **lexical ESA kernel** represents each example in the pair as the set of words that do not occur in the intersection of the two texts. The similarity is calculated as in Equation (3) with X and Y being the ESA vectors of each word from the first and second text representations, respectively.

TK - The **tree kernel** representation is based on the definition by Zanzotto et al. (2011). Briefly,

[1] http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/

each piece of text is parsed[2]; the non-terminal nodes of the parse tree, stopwords, and out-of-dictionary terms are all assigned a new random vector (Equation 1); while the leaves that occurred in the BEAGLE training corpus are assigned their learned distributional vectors (Section 2.1.1).

Each subtree of a tree is encoded recursively as a vector, where the distributional vectors representing each node are combined using the circular convolution operator (Plate, 1994; Jones and Mewhort, 2007). The whole tree is represented as a set of vectors, one for each subtree.

DK - The **dependency kernel** representation encodes each dependency pair as a separate vector, discounting the labels. The non-stopword terminals are represented as their distributional vectors, while the stopwords and out-of-dictionary terms are given a unique random vector. The vector for the dependency pair is obtained via a circular convolution of the individual word vectors.

MDK - The **multiple dependency kernel** is constructed like the dependency kernel, but similarity is calculated separately between all the the pairs that share the same dependency label. The combined similarity for all dependency labels in the parse is then calculated using least squares linear regression. While at the later stage we use GPR to combine all of the different kernels, for MDK we found that linear regression provided better performance.

DGK - The **depgram kernel** represents each dependency pair as an ESA vector obtained by searching the ESA collection for the two words in the dependency pair joined by the *AND* operator. The DGK representation only contains the dependencies that occur in one similarity text or the other, but not in both.

2.3 Regression

Each of the kernel measures above is used to calculate a similarity score between a pair of texts. The different similarity scores are then combined using Gaussian process regression (GPR) (Rasmussen and Williams, 2006). GPR is a probabilistic regression method where the weights are modelled as Gaussian random variables. GPR is defined by a covariance function, which is akin to the kernel function in the support vector machine. We used the squared exponential isotropic covariance function (also known as the radial basis function):

$$cov(x_i, x_j) = p_1^2 e^{\frac{(x_i - x_j)^T \cdot (p_2 * I)^{-1} \cdot (x_i - x_j)}{2}} + p_3^2 \delta_{ij}$$

with parameters $p_1 = 1$, $p_2 = 1$, and $p_3 = 0.01$. We found that training for parameters increased overfitting and produced worse results in validation experiments.

3 Submitted Runs

We submitted three runs. This is not sufficient for a full evaluation of the new methods we proposed here, but it gives us an inkling of general trends. To choose the composition of the submissions, we used STS 2012 training data for training, and STS 2012 test data for validation (Agirre et al., 2012). The final submitted runs also used some of the STS 2012 test data for training.

Basic - With this run we were examining if a simple introduction of syntactic structure can improve over the baseline performance. We trained a GPR combination of the linear and tree kernels (LK-TK) on the MSRpar training data. In validation experiments we found that this data set in general gave the most consistent performance for regression training.

Custom - Here we tried to approximate the best training setup for each type of data. We only had training data for OnWN and for this dataset we were able to improve over the LK-TK setup; however, the settings for the rest of the data sets were guesses based on observations from the validation experiments and overall performed poorly. OnWN was trained on MSRpar train with LK and DK. The headlines model was trained on MSRpar train and Europarl test, with LK-LEK-TK-DK-TKND-DKND-MDK (trained on Europarl).[3] FNWN was trained on MSRpar train and OnWN test with LK-LEK-DGK-TK-DK-TKND-DKND. Finally, the SMT model

[2]Because many of the datasets contained incomplete or ungrammatical sentences, we had to approximate some parses. The parsing was done using the Stanford parser (Klein and Manning, 2003), which failed on some overly long sentences, which we therefore segmented at conjunctions or commas. Since our methods only compared subtrees of parses, we simply took the union of all the partial parses for a given sentence.

[3]TKND and DKND are the versions of the tree and dependency kernels where no distributional vectors were used.

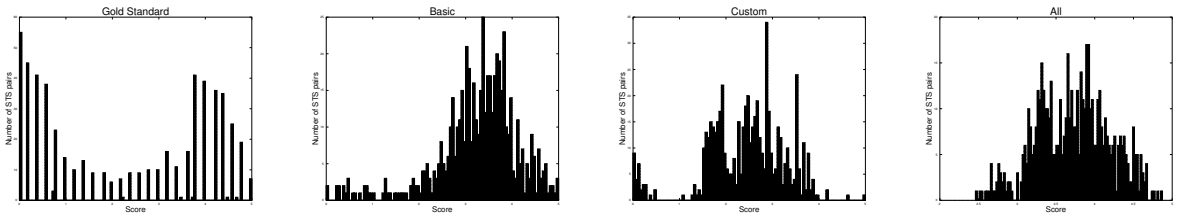

Figure 1: Score distributions of different runs on the OnWN dataset

was trained on MSRpar train and Europarl test with LK-LEK-TK-DK-TKND-DKND-MDK (trained on MSRpar).

All - As in the LK-TK experiment, we used the same model on all of the data sets. It was trained on all of the training data except MSRvid, using all eight kernel types defined above. In summary we used the LK-LEK-TK-TKND-DK-DKND-MDK-DGK kernel combination. MDK was trained on the 2012 training portion of MSRpar.

4 Discussion

From the shared task results in Table 1, we can see that Basic is our highest ranked run. It has also achieved the best performance on all data sets. The LK on its own improves slightly on the task baseline by removing stop words and using stemming, while the introduction of TK contributes syntactic and distributional information. With the Custom run, we were trying to manually estimate which training data would best reflect properties of particular test data, and to customise the kernel combination through validation experiments. The only data set for which this led to an improvement is OnWN, indicating that customised settings can be beneficial, but that a more scientific method for matching of training and test data properties is required. In the All run, we were examining the effects that maximising the amount of training data and the number of kernel measures has on the output predictions. The results show that swamping the regression with models and training data leads to overly normalised output and a decrease in performance.

While the evaluation measure, Pearson correlation, does not take into account the shape of the output distribution, Figure 1 shows that this information may be a useful indicator of model quality and behaviour. In particular, the role of the regression component in our approach is to learn a transformation from the output distributions of the models to the distribution of the training data gold standard. This makes it sensitive to the choice of training data, which ideally would have similar characteristics to the individual kernels, as well as a similar gold standard distribution to the test data. We can see in Figure 1 that the training data and choice of kernels influence the output distribution.

Analysis of the minimum, first quartile, median, third quartile, and maximum statistics of the distributions in Figure 1 demonstrates that, while it is difficult to visually evaluate the similarities of the different distributions, the smallest squared error is between the gold standard and the Custom run. This suggests that properties other than the rank order may also be good indicators in training and testing of STS methods.

	hdlns	OnWN	FNWN	SMT	mean	rank
BL	0.5399	0.2828	0.2146	0.2861	0.3639	71
Basic	0.6399	0.4440	0.3995	0.3400	0.4709	51
Cstm	0.4962	0.5639	0.1724	0.3006	0.4207	60
All	0.5510	0.3099	0.2385	0.1171	0.3200	78

Table 1: Shared task results: Pearson correlation with the gold standard

Acknowledgments

Tamara Polajnar is supported by the ERC Starting Grant, DisCoTex, awarded to Stephen Clark, and Laura Rimell and Douwe Kiela by EPSRC grant EP/I037512/1: A Unified Model of Compositional and Distributional Semantics: Theory and Applications.

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. UKP: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of the 6th International Workshop on Semantic Evaluation, held in conjunction with the 1st Joint Conference on Lexical and Computational Semantics*, pages 435–440, Montréal, Canada, June. Association for Computational Linguistics.

Danilo Croce, Paolo Annesi, Valerio Storch, and Roberto Basili. 2012. UNITOR: Combining semantic text similarity functions through sv regression. In *Proceedings of the 6th International Workshop on Semantic Evaluation, held in conjunction with the 1st Joint Conference on Lexical and Computational Semantics*, pages 597–602, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 1606–1611, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Thorsten Joachims. 1998. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142, London, UK, UK. Springer-Verlag.

Michael N. Jones and Douglas J. K. Mewhort. 2007. Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, 114:1–37.

S. Kaski. 1998. Dimensionality reduction by random mapping: fast similarity computation for clustering. In *Proceedings of the 1998 IEEE International Joint Conference on Neural Networks*, volume 1, pages 413–418 vol.1, May.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.

T. A. Plate. 1994. *Distributed Representations and Nested Compositional Structure*. Ph.D. thesis, University of Toronto.

T Polajnar, T Damoulas, and M Girolami. 2011. Protein interaction sentence detection using multiple semantic kernels. *J Biomed Semantics*, 2(1):1–1.

M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137, July.

C. E. Rasmussen and C. K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. MIT Press.

Stephen Robertson, Hugo Zaragoza, and Michael Taylor. 2004. Simple BM25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, pages 42–49, New York, NY, USA. ACM.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA.

Fabio Massimo Zanzotto and Lorenzo Dell'Arciprete. 2011. Distributed structures and distributional meaning. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*, DiSCo '11, pages 10–15, Stroudsburg, PA, USA. Association for Computational Linguistics.