A Convolution Kernel for Sentiment Analysis using Word-Embeddings

James Thorne

Department of Computer Science
University of Sheffield
jthorne1@sheffield.ac.uk

Abstract. Accurate analysis of a sentence's sentiment requires understanding of how words interact to convey emotion. Current works use the sentence's parse tree and recursively compute sentiment of the constituent phrases. This approach is expensive and requires a human to annotate all subtrees of a sentence. We examine how using a lexical similarity kernel can leverage word-embeddings, generated in an unsupervised manner, and capture the interactions between all word units. In our evaluation, we find that this type of kernel performs with comparable accuracy to the state-of-the-art for polar sentiment analysis. We examine weighting features by node-depth in the sentence's parse tree; however, this yielded unsatisfactory results. The sentence's parse tree must be leveraged to attain better sentiment analysis and we are confident that this kernel provides a framework that can be extended in future to better make use of this information.

Keywords: Natural Language Processing, Support Vector Machine, Sentiment Analysis, Kernel, Convolution, Tree Kernel, Embeddings

1 Introduction

Sentiment analysis is the task of identifying and extracting opinions and emotion from natural language. A large body of sentiment analysis work in the consumer domain been motivated by the need to understand the opinions, attitudes and emotions of customers.

A word's sentiment is often disconnected from the semantic meaning of individual word units, requiring examination of a sentence as a whole. This is highlighted with a simple example: the two sentences *"The car's steering is unpredictable"* and *"The movie's plot was unpredictable"* both use *unpredictable* as a valence shifter. However, one sentence is positive and the other is negative.

Recently there has been a large body of work focused on generation of distributional word representations called embeddings (see [1–3]): real-valued vectors which describe the meaning of word units. This both allows the similarity of words to be inferred through standard measures such as the cosine distance and allows the meaning of unseen words to be inferred through the context in which that word appears in. Embeddings abstract away from word surface forms and

have been used as features in classification tasks such as named entity recognition [4] and sentiment analysis [5].

This report investigates the modelling of sentiment analysis as a kernel-based classification task. Kernel methods are a powerful machine learning technique that allow features to be modelled in a high dimensional, implicit, metric space. We incorporate both word embeddings learned through word2vec [2] and features from the sentences parse tree in a convolution kernel that is an extension of the lexical semantic kernel [6]. The working assumption of this model is that words which appear deeper in a sentence's parse tree have less weight in the semantic orientation of the sentence.

The differentiating factor in this work is that the word embeddings are learned in an unsupervised manner from a neutral 'out-of-domain' data set. This is contrasted to the leading state-of-the-art method by Socher et al[7] which learns word embeddings in a semi-supervised environment and makes use of human-annotated sentiment labels in training the embeddings.

We find that the addition of node depth as weights in the Lexical Semantic Tree Kernel (LSTK) adversely affects the model's accuracy. This reduces the accuracy below that of other sentiment classification methods such as Recursive Neural Tensor Networks [7] and Paragraph Vector [8]. However, we find that by simply incorporating embeddings into the Lexical Semantic Kernel, we are able to achieve an accuracy score that is comparable to the vector-average of embeddings in [7] despite only using embeddings generated on an entirely out-of-domain dataset.

2 Method

2.1 Word Embeddings and Word2Vec

Word embeddings are low-rank factorisations of a word-context Pointwise Mutual Information (PMI) matrix. The inner dimensions of the factor matrices correspond to word senses observed in training. The state of the art method, Word2vec [2, 9] computes these factorised matrix components implicitly [10].

2.2 Lexical Semantic Tree Kernel

This project introduces the Lexical Semantic Tree Kernel (LSTK). The LSTK incorporates the depth of words in a sentence's constituency tree as feature weights for a Lexical Semantic Kernel (LSK) [6].

A general LSK is defined as a pairwise similarity convolution kernel over a pair of documents. To incorporate real-valued embedding vectors we alter the kernel to use the cosine similarity of embeddings instead of graph distance in an ontology.

$$K(x, y) = \sum_{w_1 \in x} \sum_{w_2 \in y} \lambda_{w_1} \lambda_{w_2} \sigma(w_1, w_2) \tag{1}$$

We define the LSTK over a pair of sentence parse trees (T_1,T_2) as the word similarity score between the words at leaf nodes (words) weighted by the node depth. This weighting was chosen under the assumption and working hypothesis that valence shifters which have a lower depth in the tree (i.e. they are closest to the root node) have a higher influence on a sentence's sentiment orientation over the nodes with a higher depth.

$$K(T_1, T_2) = \sum_{w_1 \in \text{leaf}(T_1)} \sum_{w_2 \in \text{leaf}(T_2)} \frac{1}{\text{depth}(w_1) \cdot \text{depth}(w_2)} \sigma(w_1, w_2) \qquad (2)$$

Optimisation: The distributive property of the dot-product operation is exploited to reduce the complexity of the LSK to linear with respect to the sentence lengths of x and y:

$$K(x, y) = \sum_{w_1 \in x} \sum_{w_2 \in y} \lambda_{w_1} \lambda_{w_2} \sigma(w_1, w_2) = \sigma(\sum_{w_1 \in x} \lambda_{w_1} w_1, \sum_{w_2 \in y} \lambda_{w_2} w_2) \qquad (3)$$

3 Results

LSK and LSTK kernels were tested using the Stanford Sentiment Treebank [7] dataset. These kernels were applied to the sentence-level polar and fine-grained sentiment analysis of movie reviews. For the fine-grained task, 8544 training and 2210 test samples were annotated with a continuous score which captured the strength and polarity of the sentiment; this score is quantised into five classes (`very negative`, `negative`, `neutral`, `positive` and `very positive`). We complete the polar task in the same manner as [7] to enable comparison: the `very positive` and `positive` classes were merged (likewise for the negative classes) and the `neutral` class was dropped. This reduced the number of training samples to 6920 and the number of test samples to 1821.

The evaluation was conducted using the F1-score classification metric: the harmonic mean of the classifier's precision (ratio of false positives) and recall (ratio of false negatives). In the fine-grained dataset, class numbers were not balanced (ranging between 271 and 604 samples per class). The summary statistics presented are weighted average scores based on the number of test samples.

Adding the depth of nodes as weights in the LSTK did not improve the accuracy over our baseline. This allows us to reject our working hypothesis and suggests that depth of nodes in a sentence's constituency tree alone does not play an important factor in determining a word's weight for comparing similarity of two sentences.

Neither of the new techniques was able to improve performance for the fine-grained sentiment analysis task. While the current state of the art RNTN performs with a mere 45% accuracy [7], it better captures the interplay between phrases within a sentence and presents significantly higher accuracy than the techniques presented in this work.

Method	Fine-Grained			Polar		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Bag of Words SVM [7]	-	-	0.407	-	-	0.794
RNTN [7]	-	-	0.457	-	-	0.854
VecAvg [7]	-	-	0.327	-	-	0.801
LSK	0.4989	0.4226	0.3394	0.8025	0.8018	0.8017
LSTK	0.4797	0.3898	0.3003	0.7799	0.7796	0.7795

Table 1. Sentence-level Sentiment analysis task over the Stanford Sentiment Treebank

One of the key differences between how the RNTN and VecAvg methods were trained in [7] is that the embeddings are generated in a supervised environment that exploits the human-annotated sentiment for all sub-phrases within a sentence. We contrast this against the LSK and LSTK kernel-based classifiers, which are trained only at the root level of the sentence's parse tree using embeddings that only capture semantics. While this is sufficient to generate equivalent scores for the polar task, this is not sufficient for the fine-grained analysis.

4 Conclusions

This report evaluated the performance of a kernel for sentiment analysis which weighted lexical similarity with the depth of the words in the parse tree. The working hypothesis was that words which appear deeper within a sentence's constituency tree have less weight for shifting the valence of sentences. The inclusion of this weight reduced the performance for polar sentiment analysis, leading us to reject this initial hypothesis.

The LSTK and RNTN approaches are suitable when parse trees of the sentence are available. The generation of a parse tree adds extra expense to the sentiment analysis and will introduce an additional error term. While the baseline LSK presents an advantage of not requiring a parse tree, it appears that this information is necessary to capture the finesse of the statement and improve F1 scores beyond the 'ceiling' of 0.8 for the polar task.

Even though word2vec does not incorporate sentiment into the generation of the word embeddings, using these out-of-domain unsupervised embeddings on the LSK provided satisfactory results for the polar classification. This is comparable with the baseline VecAvg without the need to train sentiment specific word embeddings.

Future work will in investigate whether sentiment-specific-word-embeddings [5] provide an increase of performance for sentiment analysis given a sentence's parse tree. Additionally, there would be merit in conducting an investigation of whether a more complex tree kernel that captures composition between different nodes rather than only evaluating leaf nodes.

References

[1] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, ACM, 2008, pp. 160–167.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[3] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: `http://www.aclweb.org/anthology/D14-1162`.

[4] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: A simple and general method for semi-supervised learning," in *Proceedings of the 48th annual meeting of the association for computational linguistics*, Association for Computational Linguistics, 2010, pp. 384–394.

[5] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification.," in *ACL (1)*, 2014, pp. 1555–1565.

[6] R. Basili, M. Cammisa, and A. Moschitti, "Effective use of wordnet semantics via kernel-based learning," in *Proceedings of the Ninth Conference on Computational Natural Language Learning*, Association for Computational Linguistics, 2005, pp. 1–8.

[7] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, Citeseer, vol. 1631, 2013, p. 1642.

[8] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," *ArXiv preprint arXiv:1405.4053*, 2014.

[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *ArXiv preprint arXiv:1301.3781*, 2013.

[10] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Advances in Neural Information Processing Systems*, 2014, pp. 2177–2185.

Acknowledgements

This report summarises work from part of an undergraduate project submitted to the Natural Language Processing module at the University of York. Thank you to Suresh Manandhar for facilitating this course and assisting with the runtime optimisation.