Evaluating the Addition of Syntactic Information in Deep Learning Models for Sentiment Analysis

Andrei Romascanu (260585208), Stefan Wapnick (260461342)

School of Computer Science, McGill University
andrei.romascanu@mail.mcgill.ca
stefan.wapnick@mail.mcgill.ca

Abstract

The use of syntactic information has been found to improve performance on different natural language processing (NLP) tasks. However, there are limited and conflicting findings on this topic in the context of deep learning for sentiment analysis. We propose a strategy to encode parse tree and part-of-speech information to augment word embeddings; and evaluate it on the large movie review dataset. Our results indicate that this syntactic information does not increase performance. Potential reasons include existing encoding of syntactic information in neural networks and word embeddings, as well as limitations with our experiment.

1 Introduction

Sentiment analysis is an important application of Natural Language Processing (NLP) with widespread use in industry, where distilled information on opinion can be a significant source of value. Typically, this task is framed as a text classification problem whereby a text document is categorized within predefined sentiment labels (Zhang, Wang, & Liu, 2018).

Over the past decade, deep learning (DL) has become a popular machine learning approach, yielding state-of-the-art results on such text classification tasks. The input features to these DL models are generally word embeddings which encode word co-occurrence information; and by extension, some degree of semantic information (Senel, Utlu, Yucesoy, Koc, & Cukur, 2017).

However, this approach ignores explicit syntactic information traditionally used in NLP, such as dependency parse trees and part-of-speech (POS) tags. There is conflicting information as to whether this information can improve performance of DL models on NLP tasks such as text classification for sentiment analysis as mentioned in our related works section.

We address this unresolved question by isolating and evaluating the impact of syntactic information on deep learning models for sentiment analysis. Specifically, we propose a novel architecture for automatically generating and encoding POS-tag and parse tree information into the input feature space of three different DL models (CNN, BiLSTM and simple feed-forward) used for polarity classification in the IMDB review dataset (Maas, et al., 2011).

2 Related Work

While the use of parse tree and POS pattern information has been successfully used to improve sentiment analysis with linguistic-based approaches (Biagioni, 2016) as well as traditional machine learning approaches (Das & Balabantaray, 2014; Nicholls & Song, 2009); there is limited research on using this information to augment the input features of DL models. Moreover, it was found that word embeddings already carry syntactic information (Andreas & Klein, 2014), and that DL models can learn internal representations that capture syntactic information (Blevins, Levy, & Zettlemoyer, 2018).

This suggests that adding syntactic information to word-embeddings as input features to deep learning models would have a limited impact on model performance. Despite this, there are findings that suggest the contrary. For example, Rezaeinia et al.

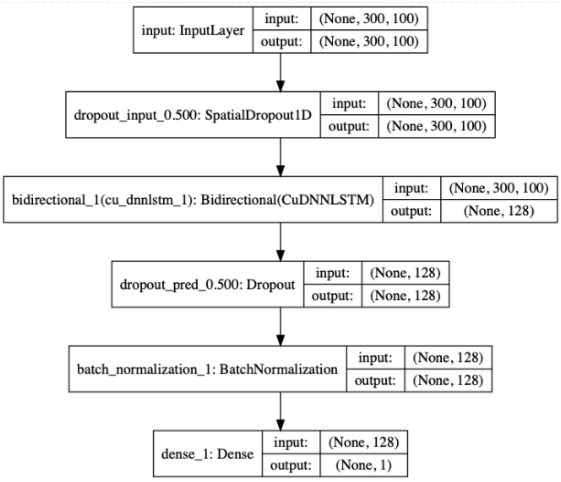

Figure 1: Baseline BiLSTM model. Other models are omitted for brevity but are included in the /baseline_architectures folder of the submission

(2018) created constant vectors for POS tags and appended them to pre-trained word-embeddings for sentiment analysis. However, their experiment does not isolate the impact of POS tag information on performance. Conversely, Liu et al. (2018) used a Bi-directional LSTM to convert a variable length sentence to a fixed-length vector representation of its parse tree structure; and found it to improve question answering on the SQuAD dataset.

We extend this research by (1) encoding POS-tags as a trainable embedding layer instead of constant vectors; (2) encoding dependency parse tree information as a filter which multiplies vectors for dependent words; (3) isolating and evaluating the impact of augmenting word embeddings with this syntactic information on a sentiment analysis task with different DL model architectures.

3 Method

3.1 Data

The large movie (imdb) review dataset (Maas, et al., 2011) was used through the Tensorflow Keras API with a train/dev/test split of 22,500/2,500/25,000 reviews. SpaCy's en_core_web_md model (Honnibal & Montani, 2017) was used to extract lemmatized word embeddings, POS tags, and dependency parse trees from reviews. To increase training speed, glove-wiki-gigaword-100 word embeddings were used through the Gensim API; as we found no significant loss in performance compared to SpaCy's larger word embeddings. Furthermore, both embeddings had a 23% out-of-vocabulary rate on the dataset lexicon.

3.2 Model Architectures

Using the Tensorflow Keras API, we created three baseline model architectures to test our hypothesis: Bidirectional LSTM (Graves, Fernández, & Schmidhuber, 2005), CNN (Kim, 2014), and feed forward. Hyper-parameter tuning was done to select values for dropout, sequence truncation length, and batch size that maximized dev-set accuracy while preventing out-of-memory errors and overfitting. The final selections are shown in **Table 1**. Furthermore, we evaluated the addition of a trainable word embedding layer with randomly initialized weights for out-of-vocabulary words for each experiment.

Baseline parameter	**Value**
Maximum text sequence length	300
Batch size	256
Dropout	0.5

Table 1: Baseline parameters

3.3 Input Features

To augment pre-trained word embeddings with POS tag information, POS tags were converted to indices using the Universal POS tags schema (Universal Dependencies Contributors, 2014) and encoded as vectors with two approaches: (1) one-hot encoded vectors (denoted as 'pos'); (2) 10-d trainable embedding layer (denoted as 'pos-embed'). The resulting POS vectors were then combined with word embeddings using a concatenation layer. Different embedding layer dimensions were tested with no observed change in performance

Dependency parse tree information was encoded as a filter matrix which rearranges a sequence of word embeddings such that the head of each word takes the place of its child. The resulting tensor is then element-wise multiplied with the original tensor such that each word embedding is multiplied by the word embedding of its head. The intuition behind this approach was to amplify "relevant" dimensions in word embeddings, where relevance is defined as co-occurrence with the head word. A similar approach to subtract head word vectors from child word vectors to encode paradigmatic relations was attempted. However, this decreased perfor-

	LSTM		CNN		FF	
	Acc.	Δ	Acc.	Δ	Acc.	Δ
baseline	84.30	0	83.06	0	78.31	0
pos	84.17	**-0.13**	82.77	**-0.28**	78.82	**0.51**
pos-embed	84.66	**0.36**	82.48	**-0.57**	78.82	**0.51**
parse	79.90	**-4.40**	76.85	**-6.20**	73.32	**-4.99**
pos + parse	79.49	**-4.80**	76.50	**-6.56**	73.19	**-5.12**
pos-embed + parse	80.97	**-3.33**	78.04	**-5.02**	74.71	**-3.60**

Table 2: Test accuracy without word-embedding training

	LSTM		CNN		FF	
	Acc.	Δ	Acc.	Δ	Acc.	Δ
baseline	86.21	0	87.25	0	86.91	0
pos	86.78	**0.57**	87.90	**0.67**	86.83	**-0.08**
pos-embed	86.70	**0.49**	87.92	**0.64**	86.85	**-0.06**
parse	83.85	**-2.36**	85.28	**-1.97**	85.4	**-1.50**
pos+ parse	84.23	**-1.98**	85.46	**-1.79**	85.51	**-1.40**
pos-embed + parse	83.18	**-3.03**	85.7	**-1.55**	85.52	**-1.39**

Table 3: Test accuracy with word-embedding training

mance, likely because paradigmatic relations captured by vector differences (Senel et al, 2017) do not correspond to the syntactic relations captured by a dependency parse tree.

3.4 Training and Evaluation

The models were trained until dev-set accuracy stopped improving to prevent overfitting. Models were then evaluated using test set accuracy, f1-score, precision, and recall. For each model architecture, a difference to the baseline was calculated to isolate and evaluate the impact of syntactic information on performance.

4 Results

A summary of the test accuracy between the augmented and baseline models are presented in **Table 2** and **Table 3**. A full set of all test run results can be found in the /results folder.

4.1 General trends

In our baseline experiments, CNN and BiLSTM architectures both outperformed FF on devsets. Validation accuracy during training was also more stable for the CNN. However, the use of a trainable embedding layer erased these differences and increased FF performance to a similar level, at the cost of observable overfitting during training. This result was similar regardless of how out-of-vocabulary vectors were initialized (zeros, random, or constant). Furthermore, these trends were also observable during evaluation on the test-set, using the non-overfitted model-checkpoint. Interestingly, training the same model with the same parameters yielded different test set results each time, with a margin of error on the order of up to 1% according to our experiments. This is likely due to the element of randomness in dropout.

4.2 Impact of POS tag information

The addition of POS tag information had no noticeable impact on any of the models during training; however the validation accuracy during training was more volatile with one-hot encodings than with trainable embedding layer encodings. Furthermore, our presented results indicate that the addition of POS tag information improved test set accuracy performance by up to 0.7%. However, this is comparable to our observed margin of error, calling into question the significance of these results and suggesting that POS tag information had no significant impact on our DL model performance, regardless of other experiment parameters.

4.3 Impact of POS tag information

In contrast, our strategy for including dependency parse tree information had a significant negative impact during training for all models: leading to slower convergence, more volatile and decreased validation accuracy. Furthermore, our presented results show that this approach caused a decrease in test accuracy of up to 5%. The negative impact was diminished by having a trainable embedding layer. Similar trends were observed for all three model architectures, indicating that this approach is detrimental regardless of the model.

5 Discussion and Conclusion

Our results suggest that our approach for adding syntactic information to the input feature space of a DL model does not increase performance in sentiment analysis; and can actually decrease it. A potential reason for this is that sentiment analysis is a relatively simple text classification task that has diminishing returns from added model/feature complexity. Indeed, this is reinforced by the fact that the simple FF neural network had accuracy on par with that

of the more complex CNN and BiLSTM architectures. However, the use of syntactic information on more complex models might better improve performance for more complex sequence labeling tasks such as question answering. Furthermore, these results might be explained by the hypothesis that neural networks and word embeddings can already learn and encode this information. Nevertheless, our results do not necessarily mean that POS tag and dependency parse tree information cannot improve performance. Indeed, there are limitations to our experiment that potentially play a role in these results.

5.1 Limitations and Potential Improvements

Inaccurate Syntactic Information:
Using machine-generated syntactic information as opposed to gold-standard information means we cannot be certain the information is accurate, despite the state-of-the-art accuracy of SpaCy's parser.
Inadequate POS Embedding:
Our approach to POS tags embeddings might not capture syntactic information adequately. While using a trainable embedding layer helped smooth validation accuracy over constant one-hot encoded vectors, the learned embeddings might be overfitting to the training data. Alternatively, the POS embeddings could be improved by unsupervised training on a larger corpus.
Inadequate Encoding of Dependency Parse:
Our approach to encoding dependency parse tree information is clearly flawed as indicated by our results. This can be attributed to the fact that we are introducing a lot of noise in our feature inputs by indiscriminately multiplying word embeddings for all dependency relations. This approach can be improved by including dependency relations in our model and learning different weights for different relations. Alternatively, instead of modifying the existing input features, the parse tree information could be encoded in separate embeddings.
Other Limitations:
The use of lemmatization in this experiment might account for some loss of information when generating word embeddings. Furthermore, our pre-trained word embeddings were trained on the Wikipedia corpus, where the language differs from the one found in movie reviews. This leads to issues such as out-of-vocabulary words and similar embeddings for sentiment-ladden antonyms. Lastly, the decision to truncate sentences for training efficiency reasons likely had a negative impact on our results, and may have decreased the efficiency of including syntactic information.

6 Statement of Contribution

6.1 Literature Review

Both team members worked jointly during the literature review process and contributed equally.

6.2 Implementation

Andrei: Data preprocessing, extraction of syntactic data, word index processing, sequence class data generator, experiment wrapper, result analysis.
Stefan: Data visualization, train/dev split, word vector embeddings cache, file system helper methods, Keras neural network model generation, result analysis.

6.3 Documentation

Andrei: Draft of introduction, related world, discussion and conclusion.
Stefan: Draft of method and result sections.

Final review and formatting were done jointly by both team members.

References

Andreas, J., & Klein, D. (2014). How much do word embeddings encode about syntax? *ACL*.

Biagioni, R. (2016). *The SenticNet Sentiment Lexicon: Exploring Semantic Richness in Multi-Word Concepts.* Springer.

Blevins, T., Levy, O., & Zettlemoyer, L. (2018). Deep RNNs Encode Soft Hierarchical Syntax. *ACL*.

Das, O., & Balabantaray, R. C. (2014). Sentiment Analysis of Movie Reviews using POS tags. *International Journal of Computer Applications*, 36-41.

Graves, A., Fernández, S., & Schmidhuber, J. (2005). Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. *International Conference on Artificla Neural Networks*, 799-804.

Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *arXiv e-prints*.

Liu, R., Hu, J., Yang, Z., & Nyberg, E. (2018). *Structural Embedding of Syntactic Trees for Machine Comprehension.* Carnegie Mellon University.

Maas, A., Daly, R., Pham, P., . Huang, D., Ng, A., & Potts, C. (2011). Learning Word Vectors for Sentiment Analysis. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (pp. 142--150). Portland: Association for Computational Linguistics.

Nicholls, C., & Song, F. (2009). Improving Sentiment Anaysis With Part-Of-Speech Weighting. *Eighth International Conference on Machine Learning and Cybernetics.* Baoding.

Rezaeinia, S. M., Rahmani, R., & Ghodsi, A. (2018). *Text Classification based on Multiple Block.*

Senel, L. K., Utlu, ˙., Yucesoy, V., Koc, A., & Cukur, T. (2017). Semantic Structure and Interpretability of Word Embeddings. *arXiv e-prints*.

Universal Dependencies Contributors. (2014). *Universal POS tags*. Retrieved from http://universaldependencies.org: http://universaldependencies.org/u/pos/

Zhang, L., Wang, S., & Liu, B. (2018). Deep Learning for Sentiment Analysis: A Survey. *arXiv e-prints*.