A Deep Neural Network Sentence Level Classification Method with Context Information

Xingyi Song and **Johann Petrak**
Department of Computer Science
University of Sheffield
Sheffield, UK
{x.song, johann.petrak}@sheffield.ac.uk

Angus Roberts
NIHR Biomedical Research Centre
Institute of Psychiatry, Psychology and Neuroscience
Kings College London
London, UK
angus.roberts@kcl.ac.uk

Abstract

In the sentence classification task, context formed from sentences adjacent to the sentence being classified can provide important information for classification. This context is, however, often ignored. Where methods do make use of context, only small amounts are considered, making it difficult to scale. We present a new method for sentence classification, Context-LSTM-CNN, that makes use of potentially large contexts. The method also utilizes long-range dependencies within the sentence being classified, using an LSTM, and short-span features, using a stacked CNN. Our experiments demonstrate that this approach consistently improves over previous methods on two different datasets.

1 Introduction

Artificial neural networks (ANN) and especially Deep Neural Networks (DNN) give state-of-the art results for sentence classification tasks. Usually, sentences are treated as separate instances for the task. However, in many situations the sentence that is the focus of classification appears in a context that can provide additional information. For example, in the below sentences from the IEMOCAP dataset, it is difficult to classify M02 as showing excitement, without the prior context:

- M01: I got it. I got accepted to U.S.C..
- F01: Oh, for real?
- M02: Yes! I just found out today. I just got the letter.

Our work is motivated by sentence classification in the text of medical records, in which complex judgements may be made across several sentences, each adding weight and nuance to a point. We believe, however, that the techniqe is more widely applicable. In order to test generalisability and to allow reproducibility, we therefore present an evaluation of the method with publicy available, non-medical corpora.

Previous work on using context for sentence classification used LSTM and CNN network layers to encode the surrounding context, giving an improvement in classification accuracy (Lee and Dernoncourt, 2016). However, the use of CNN and LSTM layers imposes a significant computational cost when training the network, especially if the size of the context is large. For this reason, the approach presented in (Lee and Dernoncourt, 2016) is explicitly intended for sequential, short-text classification.

In many cases, however, the context available is of significant size. We therefore introduce a new method, **Context-LSTM-CNN**[1], which is based on the computationally efficient FOFE (Fixed Size Ordinally Forgetting) method (Zhang et al., 2015), and an architecture that combines an LSTM and CNN for the focus sentence. The method consistently improves over results obtained from either LSTM alone, CNN alone, or these two combined, with little increase in training time.

This paper makes three contributions: 1) a demonstration of the importance of context in some sentence classification tasks; 2) an adaptation of existing datasets for such sentence classification tasks, in order to support reproducibility of evaluations; 3) a neural architecture for sentence classification that outperforms previous methods, and can include context of arbitrary size without incurring a large computational cost.

2 Related work

Since their introduction (Collobert et al., 2011), CNNs with word embedding language models have become common for text classification tasks (Kim, 2014; Conneau et al., 2017). One limitation of the original CNN approach is the loss

[1]The code is publicly available at https://github.com/deansong/contextLSTMCNN

of long distance dependencies. In order to deal with this in image and speech recognition tasks, Xu et al. (2015); Sainath et al. (2015) combined CNNs with a Recurrent Neural Network (RNN) layer. Zhou et al. (2015) subsequently applied this to text classification. However, the CNN-RNN approach was originally devised for sequence labelling, is biased towards later words in the sequence, and does not perform better than CNN alone. Huynh et al. (2016) suggested reversing the architecture to first apply the RNN followed by a CNN with pooling to obtain global features. This gave results that improved over CNN-RNN, but not over CNN alone. In this paper, we build on Huynh et al. (2016)'s approach by replacing the GRU-based RNN (Cho et al., 2014) with an LSTM (Hochreiter and Schmidhuber, 1997) and by using multiple kernel sizes and more features in the subsequent CNN layer.

Lee and Dernoncourt (2016) showed that when classifying short texts, accuracy can be boosted by adding a CNN or LSTM derived vector representation of the surrounding context. For long contexts (such as patient records which may include well over 100 sentences), however, this will incur a significant additional computational cost. In this paper, we therefore apply an adaptation of the FOFE encoding (Zhang et al., 2015) to encode context.

3 Model

The Context-LSTM-CNN model is shown in Figure 1. It is based on the following components:

1. Input layer using word embeddings to encode the words of the focus sentence.
2. Bi-directional LSTM applied to the word embeddings of the focus sentence.
3. CNN on the outputs of the LSTM.
4. FOFE applied to word embeddings of both left and right context.
5. A final output layer.

In brief, an LSTM layer is used to encode the focus sentence. This is followed by convolutional layers with small-size kernels and max-pooling to extract local features at specific points from the LSTM outputs. In addition to processing the focus sentence, we also encode the full left and right contexts using an adaptation of FOFE applied to our embeddings. This encodes any variable length context into a fixed length embedding, thus allowing us to include large contexts without rapidly increasing the computational cost. The output of the FOFE layers are then each passed through separate fully connected layers, before being concatenated and connected to output layer.

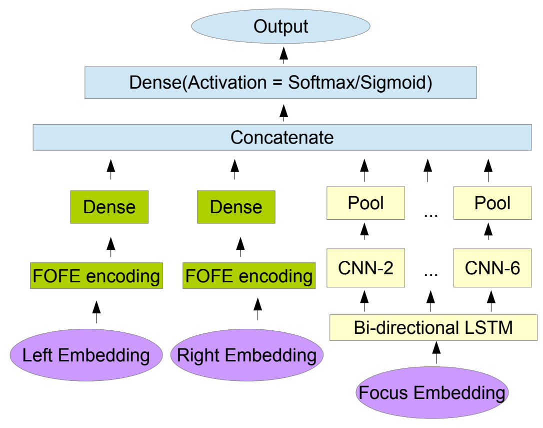

Figure 1: Structure of the C-LSTM-CNN model

In detail, the full network takes three inputs. The first is the sequence of words $X = (x_1, x_2, ...x_T)$, where T is the length of the sentence to be classified, and where each x_i is a word embedding for the respective word in this sentence. Embeddings are pre-trained by Word2Vec (Mikolov et al., 2013) on the corpus used for the respective experiment. The embeddings are not updated during the training of our network.

The second and third inputs are the left and right context, which will connect to the FOFE encoders. Each context is a sequence of sentences $X_C = (s_1, s_2, ...s_N)$, where each sentence is a sequence of word embeddings $s_n = (x_1, x_2, ...x_U)$ from the same embedding space as X.

The first component of the inputs, derived from the focus sentence, is processed by a bi-directional LSTM with one layer, in order to capture long-distance dependencies within the sentence. Since LSTMs impose a significant computational cost for very long sequences we only use this layer for the input representing the focus sentence, and not for the left and right contexts.

The LSTM generates outputs $h_{lstm} = (h_1, h_2..., h_T)$ which are passed on to the convolutional layer (CNN) in order to learn local features for different kernel sizes l from the history-aware outputs of the LSTM. For each of several kernel sizes, we generate f different features, to give CNN outputs $c^l_{cnn} = (c_1, c_2, \cdots, c_{T-l+1})$. For each CNN output c^l_{cnn}, we use max-overtime pooling to extract the most significant feature, and dropout to make the learned features more robust.

We use an adapted version of FOFE to provide information about the left and right contexts of the focus. Instead of the original 1 of k FOFE representation, we apply FOFE encoding to word2vec embeddings. This gives a weighted sum of the context word embeddings, with weights decreasing exponentially with distance from the focus.

The embedding z for a sentence $(x_1, x_2, ...x_U)$ is initialised to $z_1 = x_1$, and then calculated recursively for $u \in 2 \cdots U$ as $z_u = \alpha \cdot z_{u-1} + x_u$. The parameter α is the forgetting factor, which controls how fast the weights used for words farther away from the start of the sentence diminish. This method is fast and compactly encodes the words of a sentence in a single embedding vector.

For our use of FOFE, we encode all sentences in the document to left and right of the focus sentence, in two hierarchical steps. First we encode each context sentence into a FOFE embedding z^{sent}, with a slowly-decreasing α_{sent}. Following this, the left context FOFE encodings are themselves encoded into a single context embedding using a rapidly decreasing α_{cont}. This is calculated starting with $z_1^{cont} = z_1^{sent}$ and is calculated for $m \in 2 \cdots |C_{left}|$ as $z_m^{cont} = \alpha_{cont} \cdot z_{m-1}^{cont} + z_m^{sent}$. The right context FOFE encodings are encoded in the same way, starting with $z_{|C_{right}|}^{cont} = z_{|C_{right}|}^{sent}$ and recursively applying the same formula for $m \in |C_{right}| \cdots 2$. This gives a heavy bias towards sentences more local to the focus sentence, but only slightly decreases the importance of words within each sentence. The final FOFE embeddings for the left and right contexts are then put through a dense linear layer to obtain the hidden layer outputs, which are combined with the LSTM-CNN outputs. The concatenated outputs from the dense FOFE layers and from the CNN layer for all kernel sizes are then used as input to a final softmax output layer.

4 Experiments

We compare the performance of four different network architectures: 1) CNN only; 2) LSTM only; 3) LSTM-CNN; 4) LSTM context encoded LSTM-CNN (L-LSTM-CNN), in which the one left and right context sentence are encoded by LSTM; and 5) Context-LSTM-CNN (C-LSTM-CNN). We use the following two datasets for evaluation:

Interactive Emotional Dyadic Motion Capture Database (Busso et al., 2008)[2] (IEMOCAP). Originally created for the analysis of human emotions based on speech and video, a transcript of the speech component is available for NLP research. Each sentence in the dialogue is annotated with one of 10 types of emotion. There is a class imbalance in the labelled data, and so we follow the approach of (Chernykh et al., 2017), and only use sentences classified with one of four labels ('Anger', 'Excitement', 'Neutral' and 'Sadness'). For this dataset, instead of using left and right contexts, we assign all sentences from one person to one context and all sentences from the other person to the other context. While only the sentences with the four classes of interest are used for classification, all sentences of the dialog are used as the context. This results in a set of 4936 labelled sentences with average sentence length 14, and average document length is 986.

Drug-related Adverse Effects (Gurulingappa et al., 2012)[3] (ADE). This dataset contains sentences sampled from the abstracts of medical case reports. For each sentence, the annotation indicates whether adverse effects of a drug are being described ('Positive') or not ('Negative'). The original release of the data does not contain the document context, which we reconstructed from PubMed[4]. Sentences for which the full abstract could not be found were removed, resulting in 20,040 labelled sentences, with average sentence length 21 and average document length 129.

Model	IEMOCAP	ADE	time(s)
CNN only	58.16 (0.78)	89.49 (0.75)	218
LSTM only	56.30 (2.16)	89.04 (0.75)	648
LSTM-CNN	59.43 (1.60)	89.86 (1.06)	1239
L-LSTM-CNN	63.84 (2.03)	90.22 (0.75)	1800
C-LSTM-CNN	**71.49** (2.32)	**90.85** (0.37)	1243

Table 1: Average test accuracy and training time. Best values are marked as bold, standard deviations in parentheses

In all experiments, five-fold cross validation was used for evaluation (for comparison with (Huynh et al., 2016)). For each fold, 50 epochs were run for training using a minibatch size of 64 for each fold, and the Adamax optimization algo-

[2] http://sail.usc.edu/iemocap/iemocap_release.htm

[3] https://sites.google.com/site/adecorpus/home/document

[4] https://www.ncbi.nlm.nih.gov/pubmed/

F-measure	Anger (1,103)	Sadness (1,084)	Neutral (1,708)	Excitement (1,041)	Negative(14,854)	Positive(5,186)
CNN only	67.44 (1.02)	56.92 (3.25)	54.93 (3.70)	53.93 (2.50)	80.59 (1.08)	92.28 (0.59)
LSTM only	65.07 (2.49)	54.21 (4.03)	55.12 (2.95)	49.75 (1.80)	80.25 (1.23)	92.24 (0.54)
LSTM-CNN	67.74 (1.11)	55.86 (6.56)	57.17 (3.27)	56.95 (4.06)	81.55 (0.99)	93.00 (0.87)
L-LSTM-CNN	72.83 (1.81)	60.35 (4.65)	61.67 (3.18)	61.30 (2.64)	82.29 (0.80)	93.24 (0.60)
C-LSTM-CNN	**79.54**(1.70)	**66.07**(4.65)	**67.54**(4.72)	**73.11**(4.09)	**83.11**(0.24)	**93.72** (0.33)

Table 2: Average test F-measure for each class. Instance numbers in parentheses after class name. Best values are marked as bold, standard deviations in parentheses

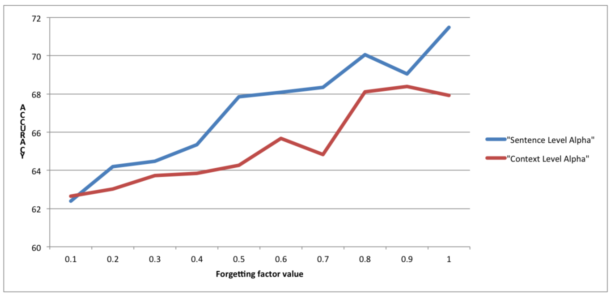

Figure 2: Context level (red line) and sentence level (blue line) forgetting factor test

rithm. To deal with label imbalance in the data, class weights w_i for class i were set proportional to $\max(f_i)/f_i$ where f_i is the frequency of class i.

We used word2vec embeddings with 50 dimensions (suggesed as sufficient by (Lai et al., 2016)). For the LSTM, 64 hidden units were used. For the CNN, layers for kernel sizes 2 to 6 were included in the network, and 64 features were used for each.

4.1 Effect of Forgetting Factors

We examined the effect of the two context encoder hyperparameters: α_{cont} (context level forgetting factor) and α_w (sentence level forgetting factor) on classification performance over the IEMOCAP dataset. We tested both in the range of 0.1 to 1 with an incremental step of 0.1. Results are shown in Figure 2. Accuracy improves as α_{cont} increases, but drops at $\alpha_{cont} = 1$, at which point all context sentence are given equal weight. This may be because context closest to the focus sentence is more important than distant context. Therefore, we select $\alpha_{cont} = 0.9$ in all experiments.

For α_{sent}, performance always increases as α_{sent} increases, with best results at $\alpha_{sent} = 1$, at which point all words in the sentence contribute equally in the context code. This implies that for individual sentences in the context, it is more preferable to lose word order, than to down weight any individual word. In all experiments, we therefore set the sentence level forgetting factor to $\alpha_{sent} = 1$

4.2 Evaluation Results

Table 1 shows the mean and standard deviations for accuracy over the cross validation folds, and training time, for both data sets. CNN alone performs better than LSTM alone in both tasks. The combined LSTM-CNN network consistently improves performance beyond both CNN alone and LSTM alone. Both context based models (L-LSTM-CNN and C-LSTM-CNN) perform better than non context based models, but note that L-LSTM-CNN increases training time by approximately 1.5x, whereas C-LSTM-CNN shows only a marginal increase in training time, with a large increase in accuracy on the IEMOCAP corpus.

Table 2 shows the F1-measure for each class in the two datasets. Again, Context-LSTM-CNN outperforms the other models on all classes for all data sets. C-LSTM-CNN improves on average by 6.28 over L-LSTM-CNN, 10.16 over LSTM-CNN, 11.4 over CNN and 13.29 over LSTM.

We conducted a t-test between L-LSTM-CNN and C-LSTM-CNN. On IEMOCAP, C-LSTM-CNN is significantly better than L-LSTM-CNN ($p = 0.002$). On ADE, C-LSTM-CNN is not significantly better than L-LSTM-CNN ($p = 0.128$). This may because ADE sentences are less context dependent. Alternatively, as the ADE task is relatively easy, with all models able to achieve about 90% accuracy, a context based approach might not be able to further improve the accuracy.

5 Conclusion

In this paper we introduced a new ANN model, **Context-LSTM-CNN**, that combines the strength of LSTM and CNN with the lightweight context encoding algorithm, FOFE. Our model shows a consistent improvement over either a non-context based model and a LSTM context encoded model, for the sentence classification task.

Acknowledgements

This work was partially supported by the European Union under grant agreement No. 654024 SoBigData.

References

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. 2008. Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42(4):335.

Vladimir Chernykh, Grigoriy Sterling, and Pavel Prihodko. 2017. Emotion recognition from speech with recurrent neural networks. *arXiv preprint arXiv:1701.08071*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Dzmitry Bahdanau, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *The 2014 Conference on Empirical Methods in Natural Language Processing*.

Ronan Collobert, Jason Weston, Lon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*.

Alexis Conneau, Holger Schwenk, Loc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *Journal of biomedical informatics*, 45(5):885–892.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Trung Huynh, Yulan He, Allistair Willis, and Stefan Rueger. 2016. Adverse drug reaction classification with deep neural networks. In *Proceedings of The 26th International Conference on Computational Linguistics*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *The 2014 Conference on Empirical Methods in Natural Language Processing*.

S. Lai, K. Liu, S. He, and J. Zhao. 2016. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Tara Sainath, Oriol Vinyals, Andrew Senior, and Hasim Sak. 2015. Convolutional, long short-term memory, fully connected deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *Proceedings of 2015th International Conference on Machine Learning*.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. 2015. A c-lstm neural network for text classification. *arXiv151108630Z*.