DataSEARCH at IEST 2018: Multiple Word Embedding based Models for Implicit Emotion Classification of Tweets with Deep Learning

Yasas Senarath
University of Moratuwa,
Sri Lanka
wayasas.13@cse.mrt.ac.lk

Uthayasanker Thayasivam
University of Moratuwa,
Sri Lanka
rtuthaya@cse.mrt.ac.lk

Abstract

This paper describes an approach to solve implicit emotion classification with the use of pre-trained word embedding models to train multiple neural networks. The system described in this paper is composed of a sequential combination of Long Short-Term Memory and Convolutional Neural Network for feature extraction and Feedforward Neural Network for classification. In this paper, we successfully show that features extracted using multiple pre-trained embeddings can be used to improve the overall performance of the system with Emoji being one of the significant features. The evaluations show that our approach outperforms the baseline system by more than 8% without using any external corpus or lexicon. This approach is ranked 8^{th} in Implicit Emotion Shared Task (IEST) at WASSA-2018.

1 Introduction

Emotion classification is a major area of interest within the field of Sentiment Analysis (SA). Social media is a great source of emotional content since people are willing to publish their views on them. Twitter is one such platform which enables users to publish micro-blogs otherwise known as Tweets. Although, the tweets are limited by the number of characters, when viewed as a group it can be very significant. Every day, on average, around 500 million tweets are tweeted on Twitter. This has attracted much interest from both academia and industries to study about opinions in tweets.

Tweets can generally be considered to contain textual content. However, tweet text is usually informal containing much casual forms and emoji, thus bringing challenges in research.

Implicit emotions play a major challenge in emotion identification process in tweets. This is due to the informal nature of the tweet and lack of methods to properly model such sentences. Here the term "implicit emotion" can be defined as the emotion conveyed in the text without stating the words denoting the emotion directly.

There is an effect of implicit emotions on opinion analysis tasks such as emotion identification and emotional intensity prediction. However, techniques for modeling implicit emotions in tweets lack the sufficient performance. Therefore, this study makes a major contribution to research by exploring methods for properly modeling a tweet.

Implicit Emotion Shared Task (IEST) (Klinger et al., 2018) hosted by WASSA-2018[1] poses a similar task of finding the emotion expressed in a tweet out of six basic emotions without the use of the word denoting the emotion. This paper presents our approach to solve the above problem. We were ranked 8^{th} in the competition related to this task.

Artificial Neural Networks (ANN) has shown to perform better than conventional machine learning algorithms and has been used in variety of Natural Language Processing tasks (Young et al., 2017). One of the primary objectives of using neural networks is to model the non-linear relationships in data, which is observed in textual content frequently. Up to now, a number of studies confirmed the effectiveness of neural networks as feature extractors rather than the final classifier for opinion mining. A variety of neural network classifiers has been applied to similar tasks such as emotion identification, polarity classification, and other text classification tasks. Feedforward Neural Networks (FNN), Convolutional Neural Networks (CNN) (Kim, 2014), Long Short-Term Memory (LSTM) (Tran and Cheng, 2018; Socher et al.,

[1] http://implicitemotions.wassa2018.com

Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pages 211–216

https://doi.org/10.18653/v1/P17

2013) networks are commonly used in recent related work. Furthermore, researchers have studied much complex forms of Neural Networks by combining CNN and LSTM in different ways.

The rest of the paper is organized as follows: Section 2 will provide a brief description on the dataset, Section 3 describes the system architecture, Section 4 reports the results and analysis of our system, finally we conclude our work in Section 5 along with a discussion on further improvements.

2 Dataset

The dataset is labeled based on the emotion word present in the tweet before replacing that emotion word in the text with a placeholder. The dataset is labeled for six basic emotions: Anger, Sad, Joy, Fear, Disgust and Surprise. The complete details of the dataset can be found in the task description paper (Klinger et al., 2018).

3 System Description

The system consists three different components: the preprocessor, feature extractor and classifier. In this study we considered that effective classifier trained on the training dataset could be used as a feature extractor as well. This section will be subdivided to accommodate the stated components separately.

3.1 Preprocessing

The tweets contained in the dataset are preprocessed to an extent. In the dataset, the URLs were replaced with "http://url.removed", mentions with "@USERNAME" and new lines with "[NEWLINE]". Additionally, we have performed following preprocessing on the dataset: changing target term "[#TRIGGERWORD#]" to "__trigger__" and "[NEWLINE]" to "__newline__". These changes were performed to correct the tokenization. We have used TweetTokenizer [2] available in python NLTK library for tokenization. In addition to NLTK tokenizer we evaluated our system using a dictionary based tokenizer.

3.2 Feature Extraction

A number of techniques have been developed to extract features for the classifier, some of which are trained on the dataset in order to create features explicitly. The most basic feature unit is the words. We used words to obtain the Word Vectors from multiple word embedding models trained on different corpses. Although our best performing system was based on word embeddings we developed and evaluated other features as well. In this section we will describe all the features that we have tried out.

ID	Model	Corpus	Corpus Size	Dim
TW2V	Word2Vec	Twitter	400M tweets	400
GW2V	Word2Vec	Google News	100B words	300
WFT	fastText	Wiki	16B tokens	300
WSFT	fastText	Wiki Subword	16B tokens	300
TGv	Glove	Twitter	2B tweets	200
E2V	Word2Vec	Twitter	1661 emoji	300

Table 1: Embedding Models used in Experiments

Word Vectors: Table 1 summarizes all of the word embedding models we used in our implementation. It illustrates the word embedding techniques and the dataset it is trained on and its specific features as well. Additionally, it provides an identifier which we will be using to identify that word embedding in the next sections. Tweets can be represented as a word vector using the word2vec approach (Mikolov et al., 2013). GW2V has been obtained by training Word2vec on part of Google News dataset[3]. Similarly, Godin et al. (2015) has provided a word2vec model trained on twitter dataset (TW2V)[4]. Furthermore, fastText (Joulin et al., 2016) models are trained on trained on UMBC webbase corpus and statmt.org news dataset with and without subword infomation (WSFT and WFT)[5] (Mikolov et al., 2018). Glove (Pennington et al., 2014) embedding (TGv) has been trained on twitter corpus containing two billion tweets[6]. Eisner et al. (2016) has released emoji2vec (E2V)[7] a pre-trained embedding model for all Unicode emoji. Intended means of using E2V is as an extension to GW2V.

Transfer Features: Features generated by training a neural classifier on the training dataset, obtained from the last layer (layer before the output later).

[2]https://www.nltk.org/api/nltk.tokenize.html

[3]https://code.google.com/archive/p/word2vec/

[4]https://www.fredericgodin.com/software/

[5]https://fasttext.cc/docs/en/english-vectors.html

[6]https://nlp.stanford.edu/projects/glove/

[7]https://github.com/uclmr/emoji2vec

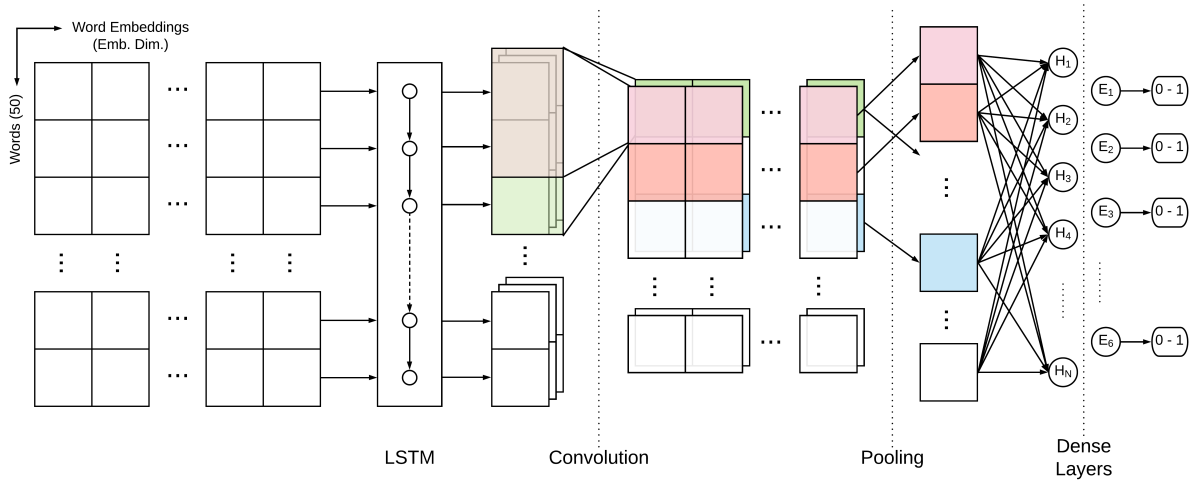

Figure 1: High-level LSTM-CNN Architecture

Section	Parameter	Value
LSTM	Num. of units	250
CNN	Num. of filters	350
	Kernel Sizes	2, 3, 5
Pooling	Method	Max
Dense Layer	Num. of units	50
	Activation	ReLU
Output Layer	Num. of Units	6
	Activation	Softmax

Table 2: Network Parameters for LSTM-CNN

3.3 Classifiers

The trial data provided in the competition is reasonably large for evaluating the model performance. As described in Section 3.2, different combinations of feature extractors were used. Following the feature extraction process, extracted features were used to train various neural networks.

3.3.1 LSTM-CNN

Two of the commonly used techniques to model text documents are Convolutional Neural Networks (CNN) and Long short-term memory (LSTM) networks. Rather than developing the neural network with CNN and LSTM separately, the proposed system is developed using a combination of CNN and LSTM. Figure 1 illustrates the proposed LSTM-CNN architecture. The hyper parameters selected for this network are tabulated in Table 2.

The network parameters are learned by optimizing the categorical cross-entropy between actual and predicted category. Optimization is performed through back propagation via mini-batch gradient descent. A batch size of 256 was used with 5 epochs to train the network. Furthermore, a dropout layer with dropout rate 0.2 is used before the dense layer when training. Adam optimization algorithm (Kingma and Ba, 2014) is used in this study for optimization . We have trained and evaluated the system with each of the word embedding models stated in Table 1.

Section	**Parameter**	**Value**
Hidden Layer 1	Num. of Units	50
	Activation	ReLU
Hidden Layer 2	Num. of Units	25
	Activation	ReLU

Table 3: Network parameters for FNN

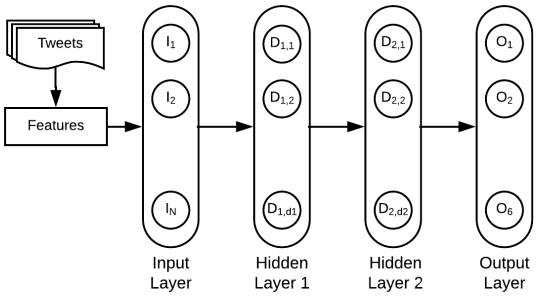

Figure 2: High-level FNN Architecture

3.3.2 Feed-forward neural network

Previous studies has shown that feed-forward neural network (FNN), can be used for modeling text

ID	Features	Trial Set			Test Set		
		Macro Precision	Macro Recall	Macro F_1	Macro Precision	Macro Recall	Macro F_1
M_{TW2V}	$TW2V$	**65.9**	**65.5**	**65.5**	**67.1**	**67.0**	**67.0**
M_{E2V}	$GW2V + E2V$	63.7	63.6	63.6	65.6	65.1	65.2
M_{GW2V}	$GW2V$	64.4	62.6	62.9	65.4	63.7	63.8
M_{WTF}	WTF	65.3	64.1	64.3	65.5	65.1	65.2
M_{WSTF}	$WSTF$	62.5	62.0	62.0	63.9	62.2	62.5
M_{TGv}	TGv	63.4	63.2	63.2	63.9	63.9	63.9
Baseline		60.1	60.1	60.1	-	-	59.8

Table 4: Evaluation of LSTM-CNN for different word embeddings

Features	Macro Precision	Macro Recall	Macro F_1
$F(M_{TW2V}) +\!\!+ F(M_{E2V})$	68.0	67.8	67.8
$F(M_{TW2V}) +\!\!+ F(M_{WTF})$	67.9	67.8	67.8
$F(M_{E2V}) +\!\!+ F(M_{WTF})$	67.1	66.7	66.8
$F(M_{E2V}) +\!\!+ F(M_{TW2V}) +\!\!+ F(M_{WTF})$	**68.3**	**68.1**	**68.1**
Baseline	-	-	59.8
IEST@WASSA 2018 Best	-	-	71.45

Table 5: Results of FNN for different feature combinations

documents (Bengio et al., 2003). Furthermore, Tang et al. (2014) has used deep neural network for learning sentiment-specific word embedding.

The proposed architecture of FNN is shown in Figure 2 and related hyper-parameters used in final system are provided in Table 3.

Training parameters of the FNN is similar to that of LSTM-CNN model. Dropout layers were used in training after each hidden layer with dropout rate of 0.5. Features used to train the FNN are transfered from dense layer of LSTM-CNN models trained with different embedding models. Several feature vectors obtained from LSTM-CNN are concatenated and provided as input to FNN. The final system used features from LSTM-CNN models trained with embeddings: TW2V, GW2V + E2V and WFT.

3.3.3 Optimization

Hyper-parameters of the neural networks should be optimized to gain better performance. They were selected based on the results on the trial set and were optimized with both manual processes and with Tree of Parzen Estimators (TPE) (Bergstra et al., 2011). However, due to the lack of processing power and time limitations we were not able to perform a comprehensive analysis on different hyper-parameter variations.

3.3.4 Implementation Details

Python is used to implement the system with Keras (Chollet et al., 2015) with Tensorflow (Abadi et al.) as the backend and Scikit-learn (Pedregosa et al., 2011) being the mostly used external libraries. Hyper-parameter optimization is performed with Hyperopt library (Bergstra et al., 2013). Any hyper-parameter not mentioned in Section 3 defaults to their default values in respective library. Furthermore, we made our source code and trained models available online [8].

4 Evaluation and Discussion

The first set of analyses examined the impact of LSTM-CNN models trained with different word embedding models. The results of the LSTM-CNN analysis are set out in Table 4. The train set evaluation is performed by training model on training dataset evaluating on trial set. Test set training data comprised of both training data and trial data.

It is apparent from this Table 4 that the model has performed similarly for both trial dataset and test dataset, achieving similar/ better F_1 scores and variations from one feature to another. We observe

[8] https://github.com/ysenarath/opinion-lab

the best performance of the system when using Word2vec trained on twitter. This could be due to the fact that it contains in-domain vocabulary. What stands out in the table is the improvement of results of M_{GW2V} with inclusion of Emoji2Vec. It can thus be suggested that Emoji provide a substantial support to finding emotion in implicit context. Furthermore, we observe that M_{WTF} performs better than M_{WSTF} and can be suggested that sub-word information provided by the embedding is not important in crating the model. Another noteworthy observation is that all the models indicated in Table 4 outperforms the baseline model in both trial and test cases, thus proving the effectiveness of the proposed model itself for implicit emotion prediction task.

In the next part of the analysis we used FNN trained using features extracted from LSTM-CNN models. Table 5 provides the evaluation results of these models on the test set. '$+\!\!+$' is used to represent vector concatenation operation and $f(M)$ denotes a function that extracts the learned features form model M from the last dense layer in the neural network for a given input text. The evaluations are performed using the three best performing LSTM-CNN models: M_{TW2V}, M_{WTF} and M_{E2V}. We have omitted M_{GW2V} for this analysis since the word vector used to train M_{GW2V} is already contained in M_{E2V}.

Results from Table 4 can be compared with the results in Table 5 which shows that the performance (precision, recall and F_1) of models in the latter has improved than the individual model variants. Closer inspection of the Table 5 shows that the best models are obtained when features from M_{TW2V} and M_{E2V} are used together. The overall best performance is obtained when features from M_{TW2V}, M_{E2V} and M_{WTF} are concatenated together.

5 Conclusion

This study is set out to propose a system for implicit emotion classification with state-of-the-art neural network classifiers. Additionally we investigate the effectiveness of combinations of different pre-trained embedding for implicit emotion classification of Tweets. In this study, a LSTM and a CNN are combined sequentially and trained with different pre-trained word embeddings to be used as a feature generator for a secondary feedforward neural network classifier to make the final classification. The results of this study indicate that the system performs well in implicit emotion identification and beats the baseline system by about 8% on the test set.

Furthermore the experiments support the idea that features extracted from several pre-trained word embedding models can be effectively combined to improve the overall classification performance . The most obvious finding to emerge from this study are that in-domain word embeddings and Emoji embeddings contribute in improving performance of implicit emotion classification. The generalisability of these results is subject to certain limitations. For instance, this research does not focus on fine-tuning the model architectures to different word-embeddings. Although this gives a general ground in comparing word-embeddings for this task, it does not provide the justification for individual capabilities. Further research will have to be conducted in order to determine the best configurations for individual word embeddings and feature combinations to improve the overall performance of the system.

Acknowledgments

The research was supported by the DataSEARCH research centre for data science, engineering, and analytics at University of Moratuwa, Sri Lanka. We thank all the contributions made by the group to this research. We would also like to thank the organizers of IEST at WASSA-2018 for organizing this shared task.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

James Bergstra, Dan Yamins, and David D Cox. 2013. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. Citeseer.

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554.

François Chollet et al. 2015. Keras. https://github.com/fchollet/keras.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *arXiv preprint arXiv:1609.08359*.

Fréderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. 2015. Multimedia lab @ acl wnut ner shared task: Named entity recognition for twitter microposts using distributed word representations. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 146–153.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Roman Klinger, Orphée de Clercq, Saif M. Mohammad, and Alexandra Balahur. 2018. Iest: Wassa-2018 implicit emotions shared task. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Brussels, Belgium. Association for Computational Linguistics.

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

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

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.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Duyu Tang, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou. 2014. Coooolll: A deep learning system for twitter sentiment classification. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 208–212.

Nam Khanh Tran and Weiwei Cheng. 2018. Multiplicative tree-structured long short-term memory networks for semantic representations. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 276–286.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2017. Recent trends in deep learning based natural language processing. *arXiv preprint arXiv:1708.02709*.