Do Specialised Affect Embeddings Improve Tweet Emotion Classification?

Zeb Goriely
zg258@cam.ac.uk
University of Cambridge
Cambridge, UK

ABSTRACT

I train two LSTM models for the task of tweet emotion classification using pre-trained word vectors for the embedding layer. I find that by concatenating specialised affect word vectors with interpretable dimensions corresponding to fine-grained emotion information, I achieve a significant improvement in classification, with the final model achieving comparable F-scores to the top-performing models of SemEval-2018 Task 1. I also find that the same improvement can be achieved by concatenating general-purpose vector sets together instead, suggesting that having multiple embeddings is the cause of the improvement, rather than the affect information itself.

KEYWORDS

affective computing, natural language processing, neural networks, emotion classification, word embeddings

1 INTRODUCTION

The study of *sentiment analysis* is an example of the overlap between the fields of affective computing and natural language processing (NLP); given a variable-length piece of text, the task is to predict whether the text conveys positive or negative (and possibly neutral) sentiment. Such work has direct applications across the world, and machine learning methods have been developed for this task for almost two decades [17].

In recent years, the field of text-based affective computing has expanded to explore the more sophisticated task of emotion classification, a multi-class classification problem involving six, eight or more emotion categories rather than just basic two or three-class valence polarity. These emotions often come from well-known psychological models, such as Ekman's six basic emotions [5] or Plutchik's eight [20]. It is easy to see how more nuanced analysis of text can directly improve systems that perform sentiment analysis. For example if a product's reviews express negative sentiment, the company's response may be very different if those reviews express anger rather than fear or sadness.

In the present study, I explore the task of emotion classification of tweets. Twitter is a particularly popular domain for the emotion classification task, due to the availability of data, the clear use-case and the presence of semi-structured information (hashtags and emojis) that can be used as distant supervision [6, 12, 26]. Tweet emotion classification was recently a subtask of the first shared task of the International Workshop on Semantic Evaluation 2018 (SemEval-2018 Task 1), among other subtasks concerned with inferring the affective state of a person, given their tweet [14]. A successful model for this task must label tweets as either containing no (or neutral) emotion, or one or more of eleven basic emotions: anger, anticipation, disgust, fear, joy, love, optimism, pessimism, sadness, surprise and trust.

Many state-of-the-art systems that perform semantic tasks such as emotion classification make use of deep neural sentence representations, often in the form of pre-trained, general-purpose *word embeddings* that capture the semantics of words [10]. In recent years, there has been an effort to create specialised word embeddings that specifically capture the affect of words, such as those provided by `AffectVec` [22] and the `EmoTag` resources [26]. These embeddings have been shown to be useful for downstream tasks including emotion classification and emotion intensity prediction of tweets.

In this study, I explore whether these specialised word vector sets can improve the performance of a LSTM-based neural models trained for the tweet emotion classification task. I first run the model using general-purpose pre-trained word vector sets, then run the model again with the `AffectVec` and `EmoTag` vectors concatenated to these pre-trained word vectors in the embedding layer. My study is based on initial results by Raji and de Melo [22], who found that their vectors improved the performance of a similar model on an emotion intensity regression task. I find that these vectors do improve micro-averaged and macro-averaged F-scores, achieving results comparable to the top-performing systems of SemEval-2018 Task 1.

However, I also find that this improvement can also be achieved by concatenating general-purpose vector sets together, raising doubts as to whether these specialised affect vectors actually introduce meaningful information to the task, as originally suggested by Raji and de Melo.

2 BACKGROUND

2.1 Word Embeddings

Most of the top-performing systems for Task 1 of SemEval-2018 made some use of deep neural sentence representation of the tweets [14]. Popular systems for producing such representations include `word2vec` [11], `fastText` [2] and `GloVe` [18]. These systems, when trained on very large corpora, produce dense vector representations of words that capture *semantic similarity*. For instance, the cosine similarity of two vectors trained using `word2vec` indicates the level of semantic similarity between the words associated with those two vectors. These embeddings have become a popular and powerful tool in the fields of NLP and cognitive science in the last decade, often helping to produce competitive results for numerous semantic tasks, including sentiment analysis [3, 24]. A particular feature of the word embeddings produced by these popular systems is that their dimensions are not interpretable. Instead, they exist in a vector

Table 1: The pre-trained word vector sets (top) and specialised affect-based vector sets (bottom) used for this experiment.

Name	Vector Length	Vocabulary Size	Training Data
fastText	200	2M	Common Crawl (600B tokens)
GloVe	200	1.2M	Twitter (2B tweets)
word2vec	300	3M	Google News (100B tokens)
AffectVec	239	76K	Wikipedia (1.8B tokens)
EmoTag	620	400K	Twitter (20M tweets)

space where word vectors that are close to each other correspond to words with similar meaning.

While many machine learning systems will run these embedding models on specific datasets to produce embeddings that best represent the data at the heart of the task, these embedding models have also been trained on huge amounts of data (many billions of tokens) to produce pre-trained, general-purpose embeddings that can be downloaded and incorporated into machine learning systems that require semantic representations of text. Several of the top-performing systems for SemEval-2018 Task 1 made use of such vector sets, feeding them into deep recursive neural models to produce a sentence-level representation of each tweet. Such models are the focus of this study.

As there are many pre-trained vector models available for download, I select three different examples, as summarised in table 1. One vector set is compared from each popular system, `GloVe`, `fastText` and `word2vec`. I select vector sets with three different vocabulary sizes (1.2M, 2M, 3M) and trained on three different corpora (Common Crawl, tweets and Google News). I believe these provide a fair coverage of the pre-trained embedding models provided by these systems.

2.2 Specialised Affect Vectors

As the field of affective computing has turned to explore the connection between human emotions and words in the last decade, a variety of databases have been produced to help assist this research. These emotion lexica provide affective scores for a large vocabulary of tokens, of varying coverage and fidelity. An early example is EmoLex [16] which provides labels for 10,000 words with binary tags for eight emotions and two sentiments, gathered using crowdsourcing. A later example is Depechemood++ [1] which instead labels over 175k words, providing numeric scores in the range 0 to 1 for eight emotions, combining crowd-based document label information with word co-occurrence statistics. Other lexica instead provide Valence, Arousal and Dominance (VAD) scores, rather than labels for discrete emotions [13]. Besides allowing researchers to look at the emotion of words directly, such databases can also assist in downstream tasks, such as sentiment analysis and emotion classification.

Raji and de Melo [22] take a different approach to producing an emotion database, based on the popularity and success of dense word embeddings. Remarking that these word embeddings do not have interpretable dimensions, they produce the `AffectVec` affective database where each word w is associated with a vector $v_w \in [0,1]^{|\Sigma|}$ for a set Σ of 239 fine-grained emotions. The values of each element of the vector quantify the extent to which that word is associated with the emotion corresponding to that dimension. As such, the dimensions are directly interpretable. These vectors are produced by first obtaining regular word vectors, using systems such as `word2vec` or `GloVe`, and adapting the vector space such that a series of soft constraints are satisfied. Similarly to how the `word2vec` vector space reduces the cosine similarity of words used in different contexts, the vector space in `AffectVec` reduces the cosine similarity of words that express opposite sentiment polarity. They then produce interpretable vectors by taking the cosine similarity of each word in this vector space with the word vector associated with the desired emotion in Σ.

Shoeb et al. [26] also produce word vectors with interpretable dimensions for affective analysis of text, distributed as part of the EmoTag1200 resources [25]. Rather than the dimensions corresponding to emotions, however, these dimensions correspond to the 620 most frequently occurring emojis. Each vector indicates the extent to which each emoji associates with the corresponding word. These are calculated very similarly to the `AffectVec` vectors, by first training a `word2vec` model on a corpus of tweets containing emojis, then calculating the cosine similarity of each word vector with each emoji vector to produce a vector $v_w \in [0,1]^{|E|}$ for the set of 620 emojis E. Emojis have previously been shown to be an effective tool for emotion-based tasks, such as the *DeepMoji* model which used emojis as noisy labels for distant supervision and achieved state-of-the-art results for sarcasm detection, sentiment analysis, and emotion classification [6]. A summary of both the `EmoTag` and `AffectVec` vectors can be seen in table 1.

Both Raji and de Melo [22] and Shoeb et al. [26] evaluate their vector sets on the WASSA (Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis) 2017 shared task [15] which involved predicting the intensity of four basic emotions (anger, fear, joy and sadness) for a set of tweets. They both use a deep neural model based on the 2nd-ranked system that uses CNN and LSTM layers, mapping words to vectors in an embedding layer. Shoeb et al. show that this system, when using the `EmoTag` vectors as embeddings, produces comparable results to the 2nd-ranked system. Raji and de Melo take a different approach, showing that the results of this model using standard `word2vec` and `GloVe` vectors as embeddings can be significantly improved by concatenating these vectors with the `AffectVec` vectors. It is this second result that I aim to extend in this study, exploring whether concatenating the `AffectVec` and `EmoTag` vectors to standard pre-trained word embeddings can improve the performance of a model on the emotion classification task.

3 METHOD

3.1 Data

I use the SemEval-2018 Task 1 dataset to train, develop and test my emotion classification model. Specifically, I use the English dataset, with the *E-c* (emotion classification) labels. This dataset consists of 6838 tweets in the training set, 886 tweets in the development set and 3259 tweets in the test set, gathered from the Twitter API in 2016 and 2017. Tweets vary in length in the range of [1, 31] words, as seen in fig. 1.

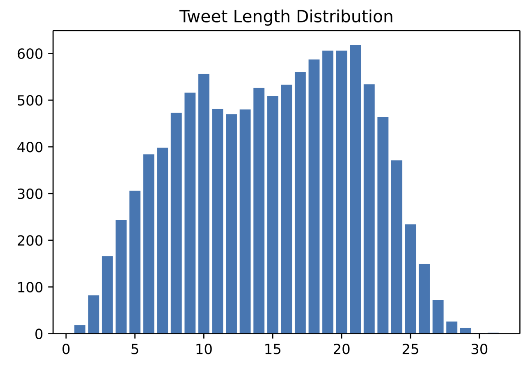

Figure 1: Histogram of tweet lengths contained in the SemEval 2018 Task 1 dataset.

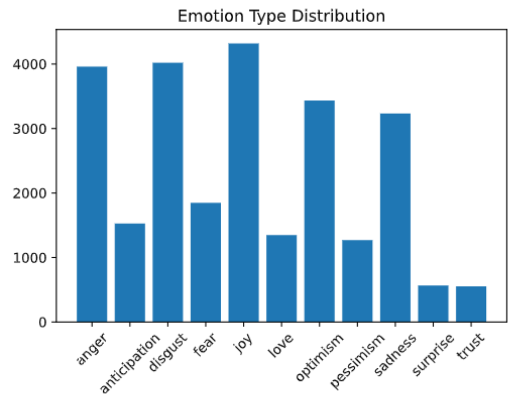

Figure 2: Number of tweets labelled with each emotion in the SemEval 2018 Task 1 dataset.

Each tweet is labelled with either no emotion label, or one or more of each of eleven different emotions. The labels are not evenly distributed, as seen in section 3.1. Labelling was performed by an average of seven annotators, and labels were selected if more than 25% of the annotators (two out of seven) agreed, a rather generous criterion. The justification for this low cutoff is that high agreement instances tend to be simple instances of each class, and are therefore easier to classify by automated systems [14]. The final labels therefore identify harder, more subtle instances of each emotion, a more appropriate difficulty for systems to be deployed in the real world.

3.2 Pre-Processing

As the dataset contains raw tweets, it must be pre-processed before being fed to the model. I tokenise each tweet, remove accents, lowercase each word and remove words longer than 15 characters or shorter than 2. This is achieved using the `gensim` package [23]. I also shuffle the tweets in each part of the dataset.

3.3 Vector Combinations

For each general-purpose pre-trained vector set, I first train and test a classification model that uses the vectors in these sets as word embeddings. I then train and test the classification model using longer word embeddings, produced by concatenating the vectors provided by `AffectVec` and `EmoTag` to the general-purpose word vectors to see whether these longer embeddings improve the performance of the classification model. This results in a comparison of nine different embeddings of different lengths. In order to ensure that any positive results are not simply a consequence of using longer embedding, I also concatenate pre-trained vector sets together as a control.

It is important to note that these vector sets have very different vocabulary sizes, as seen in table 1. The pre-trained `word2vec` vector set provides embeddings for over three million words, whereas the `AffectVec` set only has a vocabulary size of 76K. When creating embeddings by concatenating vectors from two different sets, I use the vocabulary of the first set and if a word does not appear in the second, I concatenate a zero vector of the correct length. This solution seems like it may cause frequent errors, but in reality this is a surprisingly rare occurrence. For example, 97.2% of the words in the dataset that are contained in the vocabulary of the `fastText` vector set are also contained in the vocabulary of the `AffectVec` set. This is a clear example of Zipf's law [29]: although the pre-trained word vector sets have much larger vocabularies, in practice the words they provide embeddings for are far rarer than the 76K more common words.

3.4 Classification Model

I implement two classification models, using an **LSTM** and a **Bi-LSTM** architecture. LSTMs are recurrent neural networks that includes a memory unit capable of retaining long distance dependencies. Bi-LSTMs have a similar architecture but with an additional LSTM layer that iterates over input tokens in the reverse direction. Many of the successful teams for the SemEval-2018 task used LSTM, Bi-LSTM and/or CNN models [14] and such models have been successfully applied to a variety of text classification tasks [27]. Bi-LSTMs in particular have produced state-of-the-art results for dependency parsing [8], part-of-speech tagging [19] and text classification [28].

I implement a LSTM with an embedding layer that maps each word to a vector. The length of this vector depends on the word embedding model used, as discussed in the previous section. In the case of combining the `word2vec` and `AffectVec` vectors for example, each word is mapped to a vector of length 539. These vectors are then passed to an LSTM layer with 175 units, before being fed to 11 output units with a sigmoid activation function. Each output unit corresponds to an emotion label, with output values over 0.5 indicating the presence of that label for the given input.

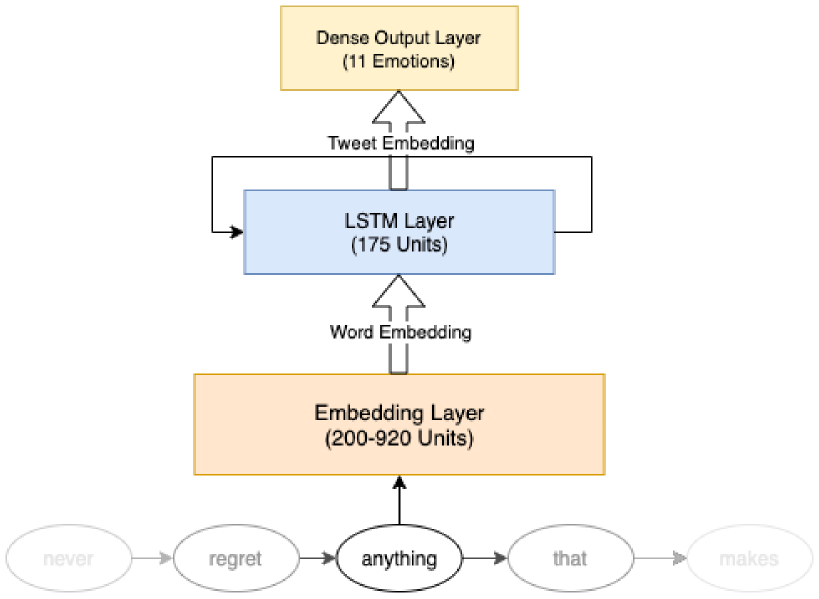

Figure 3: The LSTM model used for emotion classification. For the Bi-LSTM, there are two adjacent LSTM layers whose outputs are concatenated to provide 350 units. The number of output units from the embedding layer depend on which word embeddings are being used, for example when using the `GlOve` vectors, the embeddings will have length 200, but when appending the `word2vec` and `EmoTag` vectors, the embeddings will have length $300 + 620 = 920$. See table 1 for a full description of the different vectors used and table 2 for a list of which vector combinations are evaluated.

For regularisation, I use a dropout of 0.5 before the LSTM layer. The Bi-LSTM model has the same parameters, with the output of the two LSTM layers concatenated before being fed to the final dense output layer. A diagram of this model for the `word2vec+AffectVec` vector concatenation is given in fig. 3.

I also implement a Bi-LSTM model using the same architecture as the LSTM but with an additional LSTM layer that iterates over the tweet in the reverse direction. These two hidden layers are concatenated before being fed to the final dense output layer.

The models are trained for 8 epochs using the ADAM optimiser [7] with a batch size of 32. I used the development data provided by the dataset to tune the hyper-parameters (epoch count, number of units in each layer, and dropout rate). Hyper-parameters were tuned for the model using the `word2vec` embedding layer and kept the same for all models, altering the embedding layer for each vector combination.

The models were implemented, trained and tested using Keras [4] and the Python notebook can be found in the Github repository associated with this project[1].

4 RESULTS

4.1 Evaluation Metrics

The typical metric for measuring the performance of machine learning models is accuracy, simply calculated as the percentage of labels correctly assigned. For the multi-label case, however, this measure is not appropriate due to the distribution of labels and the fact that multiple labels can be assigned to the same input instance.

[1]https://github.com/codebyzeb/l44

Instead, I report the *F-score* achieved, averaged over each class. F-score is a derived measure related to *precision* and *recall*, two measures originating from the information retrieval literature.

Precision (P) is defined as follows:

$$P = \frac{TP}{TP + FP},$$

where TP is the number of true positives for the emotion (e.g. the number of tweets correctly predicted as “sadness”) and FP is the number of false positives (e.g. the number of tweets incorrectly predicted as “sadness”).

Recall (R) is defined as follows:

$$R = \frac{TP}{TP + FN},$$

where TP is as above, and FN is the number of false negatives (e.g. the number of tweets labelled as “sadness” in the gold standard that were not correctly labelled by the model). Each of precision and recall can usually be improved at the expense of the other. For example, classifying all the tweets as “sadness” will result in maximal recall (since all “sadness” tweets were correctly labelled) but low precision. On the other hand only labelling one tweet correctly will maximise precision, but will result in low recall. This justifies using the F-score, calculated as the harmonic mean of precision and recall[2]:

$$F\text{−score} = 2 \times \frac{P \times R}{P + R}.$$

[2]This is specifically the F_1-score, indicating that the measure gives equal weights to precision and recall. Other F_α-scores exist, giving a higher weighting to recall for higher values of α.

Table 2: Comparison of the LSTM and Bi-LSTM emotion classification model for all embedding combinations on the SemEval 2018 Task 1 dataset. F-scores are averaged over five runs and reported as percentages, with highest scores for each classifier in bold. The first nine rows report the improvement achieved by concatenating specialised affect vectors to general-purpose word vectors in the embedding layer of each model. $\underline{\text{Underlined}}$ F-scores in these rows indicate a significant improvement (t-test, $p < 0.05$) over just using the corresponding general-purpose word vector. The final three rows a control to identify whether concatenating two general-purpose vector sets together also improves performance, with $\underline{\text{underlined}}$ F-scores indicating a significant improvement over the higher-performing constituent.

	LSTM		Bi-LSTM	
Embedding	micro F-score	macro F-score	micro F-score	macro F-score
word2vec	64.3	47.0	65.4	49.5
word2vec+AffectVec	$\underline{66.2}$	$\underline{49.5}$	$\underline{66.2}$	$\underline{50.8}$
word2vec+EmoTag	$\underline{65.8}$	$\underline{49.1}$	$\underline{66.0}$	$\underline{51.0}$
GloVe	65.7	49.1	65.8	50.7
GloVe+AffectVec	$\underline{66.9}$	49.9	$\underline{\textbf{67.5}}$	$\underline{52.2}$
GloVe+EmoTag	66.4	$\underline{50.4}$	66.2	51.0
fastText	66.7	50.0	66.4	51.3
fastText+AffectVec	67.3	$\underline{51.3}$	67.3	51.8
fastText+EmoTag	66.8	51.0	66.9	52.4
word2vec+GloVe	$\underline{66.9}$	$\underline{51.3}$	$\underline{66.8}$	51.7
word2vec+fastText	66.7	$\underline{51.4}$	67.0	**52.7**
GloVe+fastText	$\underline{\textbf{67.8}}$	$\underline{\textbf{52.4}}$	$\underline{67.3}$	52.6

For each variation to the model, I report the macro-averaged F-score and the micro-averaged F-score. The former is calculated by taking the average of each measure for each class (each emotion label) and the later is similar, but the score for each class is weighted by label occurrence. These two scores indicate the performance of the model in slightly different ways, the former weights all classes equally and the latter is more representative of overall performance, taking into account the distribution of classes as a prior.

4.2 Performance of Classification Models

The F-scores achieved by the LSTM and Bi-LSTM model for all vector combinations can be seen in table 2. Both models achieve micro-averaged F-scores in the range 64.3–67.8 and macro-averaged F-scores in the range 47.0–52.7. An initial comparison between the LSTM and Bi-LSTM models reveals that both perform similarly, with the Bi-LSTM achieving slightly higher F-scores for most vector combinations. The LSTM achieves the best micro-averaged F-score of 67.8 and the Bi-LSTM achieves the best macro-averaged F-score of 52.7.

In table 3, I compare the best result achieved by the LSTM and Bi-LSTM model alongside the results achieved by the best team, the median team and the SVM baseline provided by the task [14]. With micro-average and macro-average F-scores of 67.8 and 52.4 respectively, the best LSTM model easily surpasses both the SVM baseline and the median team, and comes close to the best-performing team, with micro-averaged and macro-averaged F-scores of 70.1 and 52.8, respectively. The best performing Bi-LSTM model achieves similar scores, with 67.5 and 52.2 respectively. This validates the choice of using the LSTM and Bi-LSTM as competitive models for the task.

To understand why the micro-averaged and macro-averaged scores differ, it is useful to see the full scores achieved by a single run of one of the models. In table 4, I report the precision, recall and F-scores achieved for each of the eleven emotions by a single run of the LSTM model using the `fastText` and `AffectVec` vectors for the embedding layer. Comparing this table to the distribution of emotions in the dataset as seen in section 3.1, it is clear that the model performs far worse for the emotions less present in the dataset: anticipation, love, pessimism and trust. This results in the low macro-averaged scores seen, as every emotion is equally weighted for this calculation. When each score is weighted by the occurrence of the corresponding emotion, however, these lower individual scores have a smaller impact. This is the reason the micro-average is so much higher.

Table 3: Comparison of three models that participated in SemEval-2018 Task 1 to my best-performing LSTM and Bi-LSTM model on the same dataset. mF and MF stand for micro-averaged and macro-averaged F-score, respectively.

Model	mF	MF
SemEval 2018 Best Team	70.1	52.8
SemEval 2018 Median Team	59.9	46.4
SemEval 2018 SVM Baseline	57.0	44.3
LSTM (GloVe+fastText)	67.8	52.4
Bi-LSTM (GloVe+AffectVec)	67.5	52.2

Both scores are informative in different ways; the micro-average scores inform us about the overall performance, considering the skewed distribution of emotions in the data. In this case, the classifier performs much better on the more frequently occurring emotions, indicating a higher performance on datasets where these emotions are distributed in the same way. The macro-average scores, on

Table 4: Individual precision, recall and F-scores achieved by a single run of the LSTM model using the `fastText` and `AffectVec` vectors for the embedding layer.

Emotion	Precision	Recall	F-score
anger	80.8	74.4	77.5
anticipation	4.9	42.0	8.8
disgust	78.7	70.1	74.2
fear	65.8	83.1	73.4
joy	74.4	89.0	81.1
love	49.0	67.5	56.8
optimism	64.2	72.3	68.0
pessimism	12.0	63.4	20.2
sadness	50.4	81.9	62.4
trust	3.3	38.5	6.0
micro-average	76.4	59.8	67.1
macro-average	47.3	67.8	50.8

the other hand, indicate how the classifier will perform on any particular emotion, regardless of occurrence statistics. In this case, the lower score indicates the skewed performance; for some emotions, the classifier will perform far worse than on average.

4.3 Specialised Vector Concatenation

Looking at the first nine rows of table 2 reveals the performance of concatenating specialised affect vectors to general-purpose word vectors. In every case, the concatenation improves the micro-averaged and macro-averaged F-scores of the models by 1-2 points. With five runs of each model, most of these improvements reached significance (t-test, $p < 0.05$). This improvement is most notable for the `word2vec` vector set. Significance is reached for both F-scores across both models when the `AffectVec` and `EmoTag` vectors are concatenated to the embeddings, with F-score improvements of up to 2.5 points.

For the case of the `fastText` vectors, the highest-performing of the general purpose vectors, significance is not reached in most cases, despite the increase in F-scores in all cases. This is likely a result of the high variance between runs and the relatively smaller improvement achieved for this vector set, only about 1 point for the non-significant results compared to the 2-point gain achieved fpr the `word2vec` vectors. It is possible that averaging over more than five runs would result in significance at the level of $p < 0.05$.

Comparing the `EmoTag` and `AffectVec` vectors, both produce meaningful improvements, with the `AffectVec` vectors generally leading to greater increases. This is despite the fact that the `EmoTag` vectors are longer, have a larger vocabulary size and were trained with Tweets rather than on Wikipedia, as seen in table 1. It seems that the fine-grained emotion information provided by the `AffectVec` vectors are more useful in this task than the emoji similarity scores offered by the `EmoTag` vectors.

The global improvement provided by the `EmoTag` and `AffectVec` vectors confirms that these specialised affect vectors are indeed useful for the task of emotion classification. This follows the result of Raji and de Melo [22] who reported that their `AffectVec` vectors concatenated to general-purpose `GloVe` and `word2vec` vectors improved Pearson Correlation scores by 7-22% on a text-level emotion intensity prediction task. It is worth noting that while this may seem like a much higher improvement than achieved here, their task was a regression task for single classes, so score improvements cannot be directly compared to this classification task with eleven classes.

4.4 General Purpose Vector Concatenation

Despite this clear improvement, is it not clear whether the specific affect information provided by these vectors is the cause. It may be that any concatenation of two vector sets leads to the same improvement. To investigate this, I also concatenated the general-purpose word vector sets together and used these as the embeddings for the LSTM and Bi-LSTM model, as a control. The results of these concatenations can be seen in the last three rows of table 2. In the majority of cases, the concatenation provides a significant improvement over using either of the two vector sets individually. The `GloVe+fastText` combination even produces the micro-averaged and macro-averaged F-scores for the LSTM model. This concatenation reaches significance over using just the `fastText` vectors where the `fastText+AffectVec` and `fastText+EmoTag` combinations did not.

It is therefore clear that the concatenation of the specialised affect vectors does not provide any more of an improvement to performance for this task than just concatenating two general-purpose vector sets together. It seems that it is simply the inclusion of another feature set, rather than the specific affect information, that is the cause of this performance gain.

5 DISCUSSION

The results presented in this report indicate that specialised affect vectors do provide a significant improvement to LSTM-based models on the emotion classification task. The LSTM and Bi-LSTM models presented performed competitively to the highest performing models on Task 1 of SemEval-2018. The models from the task that performed higher also made use of features derived from affective lexica, but also used other features such as the final layers of the *DeepMoji* [6] network. Many of these systems also relied on more complicated systems for producing sentence representations, such as *Skip thoughts* and *Sentiment neurons* [9, 21]. Many systems also used a distant supervision corpus to train a neural network produce sentence embeddings, rather than just the corpus provided, as was done here. The `AffectVec` and `EmoTag` resources were not available at the time of the competition, so it is possible that their inclusion in the top-performing systems could have offered further improvements.

I also found that the same improvement can be achieved by concatenating general-purpose word vector sets together. This suggests that it is simply the inclusion of more features, rather than the specific affect information provided by these resources, that is the cause of the increased performance. This may call into question the results of Raji and de Melo [22]. It could be that the performance gain they reported could have also been achieved using general-purpose vector sets, as done here. Their model would need to be reimplemented in further work to determine if this is indeed the case.

Although affect vectors in particular may not give noticeable gains over simply concatenating general purpose vectors together, this report has shown that the concatenation itself can lead to meaningful performance gains, a noteworthy result. Many systems typically rely on embeddings trained using just one system, whether or not they use pre-trained word vectors. This result implies that such models may benefit from training embeddings on multiple datasets, using multiple systems. In the case of the emotion classification models in this paper, the concatenation of multiple vector sets leads to improvements over the individual sets in *every single case*, with the best result achieved by the `GloVe+fastText` vector combination. Further work can investigate whether such improvements could also be achieved in this manner for other semantic tasks.

CONCLUSION

In this report, I trained LSTM and Bi-LSTM models for the task of emotion classification, using the dataset from the corresponding SemEval-2018 Task 1 subtask. My final models perform competitively with the top-scoring models, achieving much higher micro-averaged and macro-averaged F-scores than the provided baseline and the median team's system. I used these two models to investigate whether performance could be improved by making use of specialised affect word vectors, by concatenating these vectors to the general-purpose word vectors used in the embedding layer of the neural networks. I found that these vectors did provide a significant increase in performance. As a control, I also appended general-purpose word vector sets together, finding that a similar and often larger performance gain was achieved. This result suggests that it is the concatenation of any vector sets, rather than the affect information, that is the cause of this performance increase, and that future models for semantic tasks could also benefit from combining multiple word representations.

REFERENCES

[1] Oscar Araque, Lorenzo Gatti, Jacopo Staiano, and Marco Guerini. 2019. Depechemood++: a bilingual emotion lexicon built through simple yet powerful techniques. *IEEE transactions on affective computing* (2019).

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[3] Sven Buechel and Udo Hahn. 2018. Word emotion induction for multiple languages as a deep multi-task learning problem. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 1907–1918.

[4] Francois Chollet et al. 2015. *Keras.* https://github.com/fchollet/keras

[5] Paul Ekman. 1999. Basic emotions. *Handbook of cognition and emotion* 98, 45-60 (1999), 16.

[6] Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524* (2017).

[7] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[8] Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics* 4 (2016), 313–327.

[9] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *arXiv preprint arXiv:1506.06726* (2015).

[10] Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior research methods, instruments, & computers* 28, 2 (1996), 203–208.

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

[12] Saif Mohammad. 2012. # Emotional tweets. 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)*. 246–255.

[13] Saif Mohammad. 2018. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 english words. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 174–184.

[14] Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of the 12th international workshop on semantic evaluation*. 1–17.

[15] Saif M Mohammad and Felipe Bravo-Marquez. 2017. WASSA-2017 shared task on emotion intensity. *arXiv preprint arXiv:1708.03700* (2017).

[16] Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational intelligence* 29, 3 (2013), 436–465.

[17] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. *arXiv preprint cs/0205070* (2002).

[18] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[19] Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv:1604.05529* (2016).

[20] Robert Plutchik. 2001. The nature of emotions: Human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice. *American scientist* 89, 4 (2001), 344–350.

[21] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444* (2017).

[22] Shahab Raji and Gerard de Melo. 2020. What Sparks Joy: The AffectVec Emotion Database. 2991–2997. https://doi.org/10.1145/3366423.3380068

[23] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, 45–50. http://is.muni.cz/publication/884893/en.

[24] Igor Santos, Nadia Nedjah, and Luiza de Macedo Mourelle. 2017. Sentiment analysis using convolutional neural network with fastText embeddings. In *2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*. IEEE, 1–5.

[25] Abu Awal Md Shoeb and Gerard de Melo. 2020. EmoTag1200: Understanding the Association between Emojis and Emotions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 8957–8967. https://www.aclweb.org/anthology/2020.emnlp-main.720

[26] Abu Awal Md Shoeb, Shahab Raji, and Gerard de Melo. 2019. EmoTag – Towards an Emotion-Based Analysis of Emojis. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. INCOMA Ltd., Varna, Bulgaria, 1094–1103. https://doi.org/10.26615/978-954-452-056-4_126

[27] Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* (2015).

[28] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639* (2016).

[29] George Kingsley Zipf. 1949. *Human behavior and the principle of least effort: An introduction to human ecology.* Addison-Wesley.