UBC-NLP at IEST 2018: Learning Implicit Emotion With an Ensemble of Language Models

Hassan Alhuzali **Mohamed Elaraby** **Muhammad Abdul-Mageed**
Natural Language Processing Lab
The University of British Columbia
{halhuzali,mohamed.elaraby}@alumni.ubc.ca,muhammad.mageeed@ubc.ca

Abstract

We describe UBC-NLP contribution to IEST-2018, focused at learning implicit emotion in Twitter data. Among the 30 participating teams, our system ranked the 4th (with 69.3% F-score). Post competition, we were able to score slightly higher than the 3rd ranking system (reaching 70.7%). Our system is trained on top of a pre-trained language model (LM), fine-tuned on the data provided by the task organizers. Our best results are acquired by an average of an ensemble of language models. We also offer an analysis of system performance and the impact of training data size on the task. For example, we show that training our best model for only one epoch with $< 40\%$ of the data enables better performance than the baseline reported by Klinger et al. (2018) for the task.

1 Introduction

Emotion is essential in human experience and communication, lending special significance to natural language processing systems aimed at learning it. Emotion detection systems can be applied in a host of domains, including health and well-being, user profiling, education, and marketing. There is a small, yet growing, body of NLP literature on emotion. Early works focused on creating and manually labeling datasets. The SemEval 2007 Affective Text task Strapparava and Mihalcea (2007) and Aman and Szpakowicz (2007) are two examples that target the news and blog domains respectively. In these works, data were labeled for the 6 basic emotions of Ekman (Ekman, 1972). More recent works exploit *distant supervision* (Mintz et al., 2009) to automatically acquire emotion data for training systems. More specifically, a number of works use hashtags like $\#happy$ and $\#sad$, especially occurring finally in Twitter data, as a proxy of emotion (Wang et al., 2012; Mohammad and Kiritchenko, 2015; Volkova and Bachrach, 2016). Abdul-Mageed and Ungar (2017) report state-of-the-art results using a large dataset acquired with hashtags. Other works exploit emojis to capture emotion carrying data (Felbo et al., 2017). Alhuzali et al. (2018) introduce a third effective approach that leverages first-person seed phrases like *"I'm happy that"* to collect emotion data.

Klinger et al. (2018) propose yet a fourth method for collecting emotion data that depends on the existence of the expression "emotion-word + one of the following words (when, that or because)" in a tweet, regardless of the position of the emotion word. In the "Implicit Emotion" shared task [1], participants were provided data representing the 6 emotions in the set (anger, disgust, fear, joy, sad, surprise). The trigger word was removed from each tweet. To illustrate, the task is to predict the emotion in a tweet like "Boys who like Starbucks make me *[#TRIGGERWORD#]* because we can go on cute coffee dates" (with the triggered word labeled as *joy*). In this paper, we describe our system submitted as part of the competition. Overall, our submission ranked the 4th out of the 30 participating teams. With further experiments, we were able to acquire better results, which would rank our model at top 3 (70.7% F-score).

The rest of the paper is organized as follows: Section 2 describes the data. Section 3 offers a description of the methods employed in our work. Section 3 is where we present our results, and we perform an analysis of these results in Section 5. We list negative experiments in Section 6 and conclude in Section 7.

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

2 Data

We use the Twitter dataset released by the organizers of the "Implicit Emotion" task, as described in the previous section. The data are partitioned into $153,383$ tweets for training, 9591 tweets for validation, and $28,757$ data points for testing. The training and validation sets were provided early for system development, while the test set was released one week before the deadline of system submission. The full details of the dataset can be found in Klinger et al. (2018). We now describe our methods in the nesxt section.

3 Methods

3.1 Pre-processing

We adopt a simple pre-processing scheme, similar to most of the pre-trained models we employ. This involves lowercasing all text and filtering out urls and user mentions. We also split clusters of emojis into individual emojis, following Duppada et al. (2018). For our vocabulary V, we retain the top $100k$ words and then remove all words occurring < 2 times, which leaves $|V| = 23,656$.

3.2 Models

We develop a host of models based on deep neural networks, using some of these models as our baseline models. As an additional baseline, we compare to Klinger et al. (2018) who propose a model based on Logistic Regression with a bag of word unigrams (BOW). All our deep learning models are based on variations of recurrent neural networks (RNNs), which have proved useful for several NLP tasks. RNNs are able to capture sequential dependencies especially in time series data, of which language can be seen as an example. One weakness of RNNs, however, lies in gradient either vanishing or exploding during training. Long-short term memory (LSTM) networks were developed to target this problem, and hence we employ these in our work. We also use a bidirectional version (BiLSTM) where the vector of representation is built as a concatenation of two vectors, one that runs from left-to-right and another running from right-to-left. Ultimately, we generate a fixed-size representation for a given tweet using the last hidden state for the Fwd and Bwd LSTM. Our systems can be categorized as follows: (1) Systems tuning simple pre-trained embeddings, (2) Systems tuning embeddings from language models, and (3) Systems directly tuning language models. We treat #1 and #2 as baseline systems, while our best models are based on #3.

3.2.1 Systems With Simple Embeddings

Character and/or Word embeddings (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2016) have boosted performance on a host of NLP tasks. Most state of the art systems now fine-tune these embeddings as a simple transfer learning technique targeting the first layer of a network (McCann et al., 2017). We make use of one such pre-trained embeddings (fastText) to identify the utility of tuning its learned weights on the task.

FastText: The first embedding model is fastText [2](Bojanowski et al., 2016), which builds representations based on characters, rather than only words, thus alleviating issues of complex morphology characetrestic of many languages like Arabic, Hebrew, and Swedish, but also enhancing representations for languages of simpler morphology like English. Additionally, fastText partially solves issues with out-of-vocabulary words since it exploits character sequences. FastText is trained on the Common Crawl dataset, consisting of 600B tokens.

For this and the next set of experiments (i.e., experiments in 3.2.2), we train both an LSTM and BiLSTM. Since we treat these as baseline systems, especially with our goal to report our experiments in available space for the competition, we try a small set of hyper-parameters, identifying best settings on our validation set. We train each network for 4 epochs each. For optimization, we use $Adam$ (Kingma and Ba, 2014). The model's weights W are initialized from a normal distribution $W \sim N$ with a small standard deviation of $\sigma = 0.05$. We apply two sources of regularization: $dropout$: we apply a dropout rate of 0.5 on the input embeddings to prevent co-adaptation of hidden units' activation, and $L2 - norm$: we also apply an L2-norm regularization with a small value (0.0001) on the hidden units layer to prevent the network from over-fitting on training set. Each of the networks has a single hidden layer. Network architectures and hyper-parameters are listed in Table 1.

3.2.2 Embedding From LMs

Peters et al. (2018) build embeddings directly from

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

Hyper-Parameter	Value
Embed-dim-fastText	300
Embed-dim-ELMo	1024
layers	1
units	300
batch size	32
epochs	4
dropout	0.5

Table 1: Network architecture and hyper-parameters for experiments with simple pre-trained embeddings with fastText 3.2.1 and ELMo 3.2.2 across our LSTM and BiLSTM networks.

language models, which they refer to as ELMo. ELMo is shown to capture both complex characteristics of words (as syntax and semantics) as well as the usage of these words across various linguistic contexts, thanks to its language modeling component. ELMo is trained on a dataset of Wikipedia and is publicly available [3], which we use as our input layer. More specifically, we extract the weighted sum of the 3 layers (word embedding, Bi-lstm-outputs1, and Bi-lstm-outputs2) and follow the same network architectures and hyper-parameters employed with fastText as we explain before.

3.2.3 Fine-Tuning LMs: ULMFiT

Another recent improvement in training NLP systems is related to the way these systems are fine-tuned, especially vis-a-vis how different layers in the network operate during training time. Howard and Ruder (2018) present **ULMFiT**[4], an example such systems that is pre-trained on a language model exploiting Wikitext-103. Ultimately, ULMFiT employs a number of techniques for training. These include "gradual unfreezing", which aims at fine-tuning each layer of the network independently and then fine-tuning all layers together. Gradual unfreezing proves useful for reducing the risk of overfitting as also found in Felbo et al. (2017). ULMFiT also uses "discriminative fine-tuning", which tunes each layer with different learning rates, the idea being different layers capture different types of information (Howard and Ruder, 2018; Peters et al., 2018). Howard and Ruder (2018) also use different learning rates, which they refer to as "slanted triangular learning rates", at different times of the training process. With ULMFiT, we experiment with different variations of LMs [5]: forward (**Fwd**), backward (**Bwd**), and an average of these (**BiLM (Fwd+Bwd)**). We follow Howard and Ruder (2018) in training each of the Fwd and Bwd models independently on the training data provided by the task organizers, and then combining their predictions using an ensemble averaging. This is the setting we refer to as BiLM. As we show in Section 3, this is a beneficial measure (similar to what Howard and Ruder (2018) also found). For our hyper-parameters for this iteration of experiments, we identify them on our validation set. We list the network architectures and hyper-parameters for this set of experiments in Table 2.

Hyper-Parameter	Value
dim-size	400
vocab	$23,656$
batches	64
layers	3
units	$1,150$
epochs	19

Table 2: Hyper-parameters for our submitted system exploiting fine-tuned language models from Howard and Ruder (2018).

4 Results

Table 3 shows results of all our models in F-score. As the Table shows, all our models achieve sizable gains over the logistic regression model introduced by (Klinger et al., 2018) as a baseline for the competition (F-score $= 60\%$). Even though our models trained based on fastText and ELMo each has a single hidden layer, which is not that deep, these at least 1.5% higher than the logistic regression model. We also observe that ELMo embeddings, which are acquired from language models rather than optimized from sequences of tokens, achieves higher performance than FastText embeddings. This is not surprising, and aligns with the results reported by Peters et al. (2018).

For results with ULMFiT, as Table 3 shows, it acquires gains over all the other models. As mentioned earlier, we experiment with different variations of LMs (Fwd, Bwd, and BiLM). Results in our submitted system are based on the Fwd model, and are at 69.4%. After system submission, we

[3]https://github.com/allenai/bilm-tf

[4]http://nlp.fast.ai/category/classification.html.

[5]Fwd and Bwd LMs are offered by the authors of the ULMFiT model (Howard and Ruder, 2018).

also experimented with the Bwd and BiLM models and were able to acquire even higher gains, putting our best performance at 70.7% (which moves us to the top 3 position).

System	Dev	Test
Baseline (Klinger et al., 2018)		
BOW Log-Reg	0.601	0.601
Embeddings		
FastText (Bojanowski et al., 2016)		
LSTM	0.629	0.629
Bi-LSTM	0.628	0.626
Embed. from LM (ELMo) (Peters et al., 2018)		
LSTM	0.635	0.635
Bi-LSTM	0.615	0.614
Fine-Tuned LM (Howard and Ruder, 2018)		
Fwd LM (submitted system)	0.694	0.693
Bwd LM	0.686	0.693
BiLM	**0.707**	**0.707**

Table 3: Results: BiLM refers to an ensemble of both the Fwd and Bwd LMs.

5 Analysis

5.1 Error Analysis

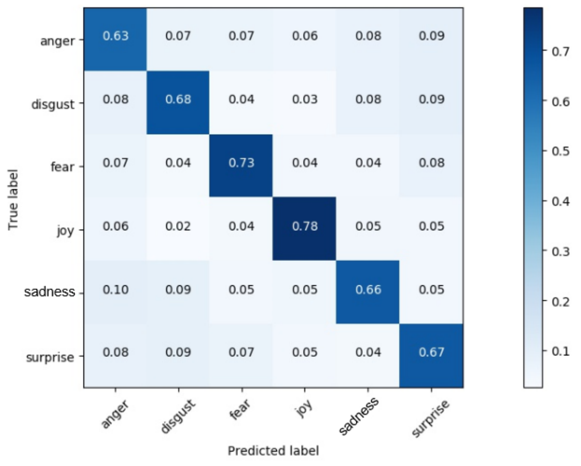

Figure 1: Confusion matrix of errors in F-score across the different emotion classes.

Using predictions from our best model (as described in Table 2), we investigate the extent with which each emotion is mislabeled and the categories with which it is confused. Figure 1 shows the confusion matrix of this analysis. As the Figure shows, *anger* is predicted with least F-score ($\% = 63$), followed by *sadness* ($\% = 66$). Figure 1 also shows that *anger* is most confused for *surprise* and *sadness* is most confused for *anger*. Additionally, *disgust* is the third most confused category ($\% = 66$), and is mislabeled as *surprise* 9% of the time. These results suggest overlap in the ways each of the emotions is expressed in the training data.

To further investigate these observations, we measure the shared vocabulary between the different classes. Figure 2 shows percentages of lexical overlap in the data, and does confirm that some categories share unigram tokens to varying degrees. Lexical overlap between classes seem to align with the error matrix in Figure 1. For example, *anger* overlaps most with *surprise* ($\% = 9$) and *sadness* overlaps most with *anger* ($\% = 10$). These findings are not surprising, since our models are based on lexical input and do not involve other types of information (e.g., POS tags). Table 4 offers examples of overlap in the form of lexical sequences between test data and training data across a number of classes.

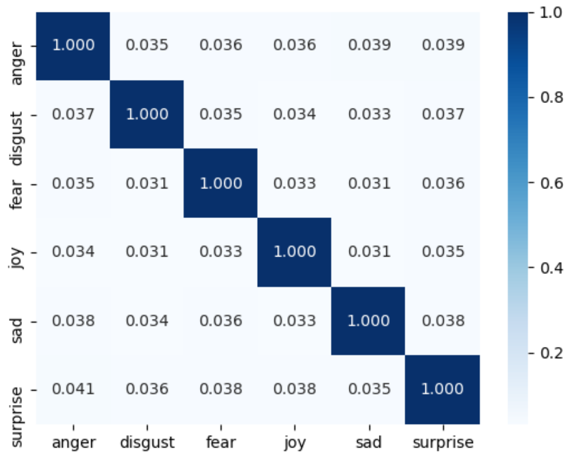

Figure 2: Heat Map for percentages of shared vocabulary between emotion classes.

5.2 Size of Training Data

We also investigate the impact of training data size on model accuracy. For this purpose, we train models with different data sizes with the best parameter settings shown in Table 2 [6]. Figure 3 shows the impact of different percentages of training examples on model performance. We test

[6]Due to the high computational cost of training these models, we only train each model with one epoch for this analysis.

Test Example	True	Predicted	Train Example	True
I'm [#TRIGGERWORD#] **that** **like none of** my friends at school have seen national lamoon's day	disgust	sad	[#TRIGGERWORD#] **that like** **none of** my videos from last night ..	sad
hey luke! I'm so [#TRIGGERWORD#] **because you** **don't follow me**, not lies, but please follow me	anger	sad	i'm so [#TRIGGERWORD#] **because** **you don't** **follow me**	sad

Table 4: Examples overlapping lexical sequences in test and training data.

model performance for this analysis on our validation data.

Interestingly, as Figure 3 shows, the model exceeds the baseline model reported by the task organizers (Klinger et al., 2018) when trained on only 10% of the training data. Additionally, the model outperforms the fastText and ELMo models by only seeing 40% of the training data. Once the model has access to 80% of the training data, its gains start to increase relatively slowly. In addition to the positive, yet unsurprising, impact that training data size has on performance, the results also reflect the utility of employing the pre-trained language model.

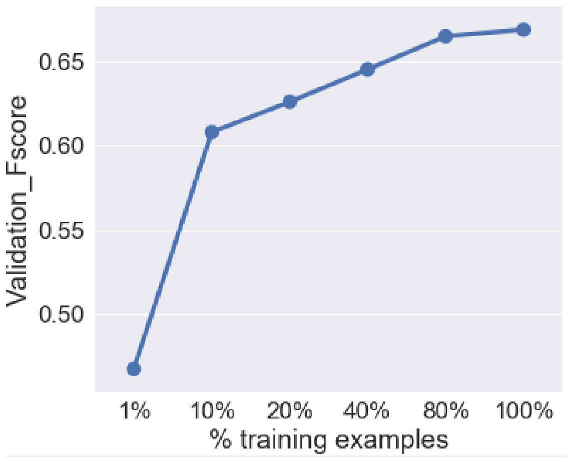

Figure 3: Impact of training data size on model performance, tested on our validation data. Results in F-score.

In order to further inspect this observation regarding the impact of language modeling, we use the same architecture reported in Table 2 to train a classifier that does not have access to the pre-trained LM. We find the classifier to achieve only 63.8 F-score. Again, this demonstrates the advantage of using the pra-trained LM.

6 Negative Experiments

We performed a number of negative experiments that we report briefly here. Our intuition is that training our models with Twitter-specific data should help classification. For this reason, we trained ULMFiT with 4.5 million tweets with the same settings reported in Table 2. We did not find this set of experiments to yield gains over the results reported in Table 3, however. For example, an Fwd LM trained on Twitter domain data yields 67.9% F-score, which is 1.4% less than the F-score obtained by the Wikipedia-trained Fwd LM in 3. The loss might be due to the smaller size of the Twitter data we train on, as compared to the Wikipedia data the ULMFiT is originally trained on (i.e., $>$ 103 million tokens).

7 Conclusion

In this paper, we described our system submitted to IEST-2018 task, focused on learning implicit emotion from Twitter data. We explored the utility of tuning different word- and character-level pre-trained representations and language modeling methods to minimize training loss. We found that the method introduced by Howard and Ruder (2018) yields best performance on the task. We note that our baselines employing sub-word embeddings (fastText) and embeddings from language models (ELMo) can be improved by using deeper neural architectures with larger model capacity, which we cast for future work. We have also shown that the classifier confuses certain emotion classes with one another, possible due to overlap of lexical sequences between training and test data. This reflects the difficulty of the task.

8 Acknowledgement

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC). The research was enabled in part by support provided by WestGrid (https://www.westgrid.ca/) and Compute Canada (www.computecanada.ca).

References

Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 718–728.

Hassan Alhuzali, Muhammad Abdul-Mageed, and Lyle Ungar. 2018. Enabling deep learning of emotion with first-person seed expressions. In *Proceedings of the Second Workshop on Computational Modeling of Peoples Opinions, Personality, and Emotions in Social Media*, pages 25–35.

Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. In *Text, Speech and Dialogue*, pages 196–205. Springer.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Venkatesh Duppada, Royal Jain, and Sushant Hiray. 2018. Seernet at semeval-2018 task 1: Domain adaptation for affect in tweets. *arXiv preprint arXiv:1804.06137*.

P. Ekman. 1972. Universal and cultural differences in facial expression of emotion. *Nebraska Symposium on Motivation*, pages 207–283.

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. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *arXiv preprint arXiv:1801.06146*.

Diederik 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.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Saif M Mohammad and Svetlana Kiritchenko. 2015. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence*, 31(2):301–326.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 70–74. Association for Computational Linguistics.

Svitlana Volkova and Yoram Bachrach. 2016. Inferring perceived demographics from user emotional tone and user-environment emotional contrast. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL*.

Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P Sheth. 2012. Harnessing twitter" big data" for automatic emotion identification. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*, pages 587–592. IEEE.