IIT Gandhinagar at SemEval-2019 Task 3: Contextual Emotion Detection Using Deep Learning

Arik Pamnani[*], **Rajat Goel**[*], **Jayesh Choudhari**, **Mayank Singh**
IIT Gandhinagar
Gujarat, India
{arik.pamnani,rajat.goel,choudhari.jayesh,singh.mayank}@iitgn.ac.in

Abstract

Recent advancements in Internet and Mobile infrastructure have resulted in the development of faster and efficient platforms of communication. These platforms include speech, facial and text-based conversational mediums. Majority of these are text-based messaging platforms. Development of *Chatbots* that automatically understand latent emotions in the textual message is a challenging task. In this paper, we present an automatic emotion detection system that aims to detect the emotion of a person textually conversing with a chatbot. We explore deep learning techniques such as CNN and LSTM based neural networks and outperformed the baseline score by 14%. The trained model and code are kept in public domain.

1 Introduction

In recent times, text has become a preferred mode of communication (Reporter, 2012; ORTUTAY, 2018) over phone/video calling or face-to-face communication. New challenges and opportunities accompany this change. Identifying sentiment from text has become a sought after research topic. Applications include detecting depression (Wang et al., 2013) or teaching empathy to chatbots (WILSON, 2016). These applications leverage NLP for extracting sentiments from text. On this line, SemEval Task 3: EmoContext (Chatterjee et al., 2019) challenges participants to identify Contextual Emotions in Text.

Challenges: The challenges with extracting sentiments from text are not only limited to the use of slang, sarcasm or multiple languages in a sentence. There is also a challenge which is presented by the use of non-standard acronyms specific to individuals and others which are present in the task's dataset 2.

[*]Equal Contribution

Existing work: For sentiment analysis, most of the previous year's submissions focused on neural networks (Nakov et al., 2016). Teams experimented with Recurrent Neural Network (RNN) (Yadav, 2016) as well as Convolutional Neural Network (CNN) based models (Ruder et al., 2016). However, some top ranking teams also used (Giorgis et al., 2016) classic machine learning models. Aiming for the best system, we started with classical machine learning algorithms like Support Vector Machine (SVM) and Logistic Regression (LR). Based on the findings from them we moved to complex models using Long Short-Term Memory (LSTM), and finally, we experimented with CNN in search for the right system.

Our contribution: In this paper, we present models to extract emotions from text. All our models are trained using only the dataset provided by EmoContext organizers. The evaluation metric set by the organizers is micro F1 score (**referred as score in rest of the paper**) on three {*Happy, Sad, Angry*} out of the four labels. We experimented by using simpler models like SVM, and Logistic regression but the score on dev set was below 0.45. We then worked with a CNN model and two LSTM based models where we were able to beat the baseline and achieve a maximum score of 0.667 on test set.

Outline: Section 2 describes the dataset and preprocessing steps. Section 3 presents model description and system information. In the next section (Section 4), we discuss experimental results and comparisons against state-of-the-art baselines. Towards the end, Section 5 and Section 6 conclude this work with current limitations and proposal for future extension.

2 Dataset

We used the dataset provided by Task 3 in SEMEVAL 2019. This task is titled as *'EmoContext: Contextual Emotion Detection in Text'*. The dataset consists of textual dialogues i.e. a user utterance along with two turns of context. Each dialogue is labelled into several emotion classes: *Happy, Sad, Angry* or *Others*. Figure 1 shows an example dialogue.

Turn 1: N u
Turn 2: Im fine, and you?
Turn 3: I am fabulous 😎

Figure 1: Example textual dialogue from EmoContext dataset.

Table 1 shows the distribution of classes in the EmoContext dataset. The dataset is further subdivided into train, dev and test sets. In this work, we use training set for model training and dev set for validation and hyper-parameter tuning.

	Others	**Happy**	**Sad**	**Angry**
Train	14948	4243	5463	5506
Dev	2338	142	125	150
Test	4677	284	250	298

Table 1: Dataset statistics.

Preprocessing: We leverage two pretrained word embedding: (i) GloVe (Pennington et al., 2014) and (ii) sentiment specific word embedding (SSWE) (Tang et al., 2014). However, several classes of words are not present in these embeddings. We list these classes below:

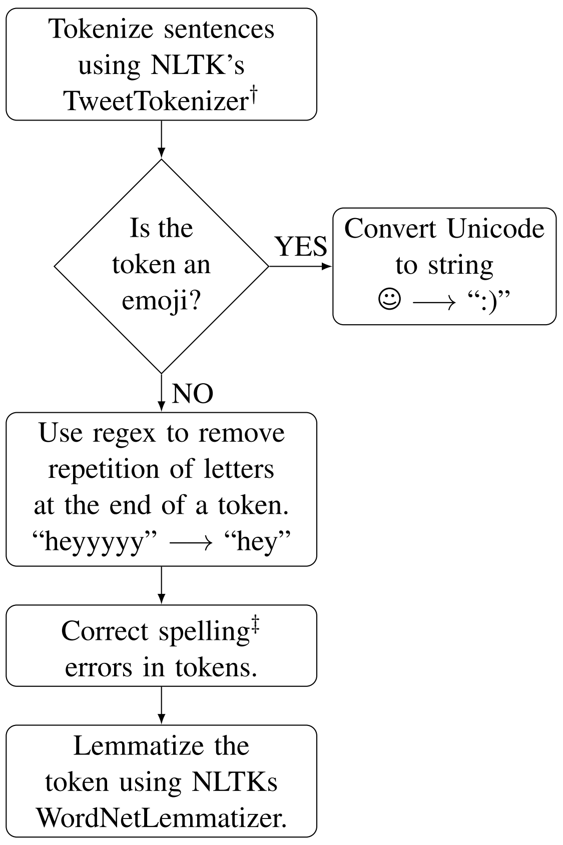
- **Emojis:** 😎, 😇, etc.
- **Elongated words:** Wowwww, noooo, etc.
- **Misspelled words:** ofcorse, activ, doin, etc.
- **Non-English words:** Chalo, kyun, etc.

We follow a standard preprocessing pipeline to address the above limitations. Figure 2 describes the dataset preprocessing pipeline.

By using the dataset preprocessing pipeline, we reduced the number of words not found in GloVe embedding from *4154* to *813* and in SSWE from *3188* to *1089*.

[†]NLTK Library (Bird et al., 2009)

[‡]For spell check we the used the following PyPI package - pypi.org/project/pyspellchecker/.



Tokenize sentences using NLTK's TweetTokenizer[†]
Is the token an emoji?
YES
Convert Unicode to string ☺ ⟶ ":)"
NO
Use regex to remove repetition of letters at the end of a token. "heyyyyy" ⟶ "hey"
Correct spelling[‡] errors in tokens.
Lemmatize the token using NLTKs WordNetLemmatizer.

Figure 2: Data processing pipeline.

3 Experiments

We experiment with several classification systems. In the following subsections we first explain the classical ML based models followed by Deep Neural Network based models.

3.1 Classical Machine Learning Methods

We learn[§] two classical ML models, (i) Support Vector Machine (SVM) and (ii) Logistic Regression (LR). The input consists a single feature vector formed by combining all sentences (turns). We term this combination of sentences as *'Dialogue'*.

We create feature vector by averaging over d dimensional GloVe representations of all the words present in the dialogue. Apart from standard averaging, we also experimented with tf-idf weighted averaging. The dialogue vector construction from tf-idf averaging scheme is described below:

$$Vector_{dialogue} = \frac{\Sigma_{i=1}^{N}(\text{tf-idf}_{w_i} \times GloVe_{w_i})}{N}$$

[§]We leverage the Scikit-learn (Pedregosa et al., 2011) implementation.

Here, N is the total number of tokens in a sentence and w_i is the i^{th} token in the dialogue. Empirically, we found that, standard averaging shows better prediction accuracy than tf-idf weighted averaging.

3.2 Deep Neural Networks

In this subsection, we describe three deep neural architectures that provide better prediction accuracy than classical ML models.

3.2.1 Convolution Neural Network (CNN)

We explore a CNN model analogous to (Kim, 2014) for classification. Figure 3 describes our CNN architecture. The model consists of an embedding layer, two convolution layers, a max pooling layer, a hidden layer, and a softmax layer. For each dialogue, the input to this model is a sequence of token indices. Input sequences are padded with zeros so that each sequence has equal length n.

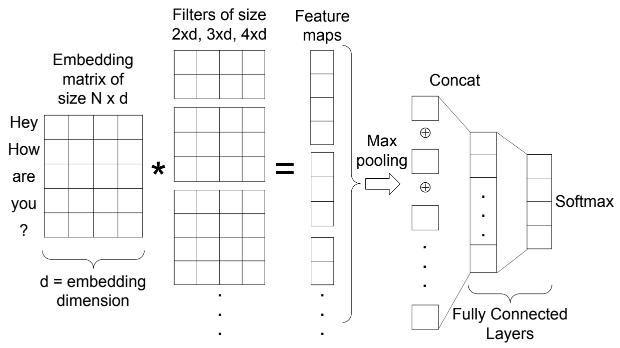


Figure 3: Architecture of the CNN model.

The embedding layer maps the input sequence to a matrix of shape $n \times d$, where n represents n^{th} word in the dialogue and d represents dimensions of the embedding. Rows of the matrix correspond to the GloVe embedding of corresponding words in the sequence. A zero vector represents words which are not present in the embedding.

At the convolution layer, filters of shape $m \times d$ slide over the input matrix to create feature maps of length $n - m + 1$. Here, m is the 'region size'. For each region size, we use k filters. Thus, the total number of feature maps is $m \times k$. We use two convolution layers, one after the other.

Next, we apply a max-pooling operation over each feature map to get a vector of length $m \times k$. At the end, we add a hidden layer followed by a softmax layer to obtain probabilities for classification. We used Keras for this model.

3.2.2 Long Short-Term Memory-I (LSTM-I)

We experiment with two Long Short-term Memory (Hochreiter and Schmidhuber, 1997) based approaches. In the first approach, we use an architecture similar to (Gupta et al., 2017) Here, similar to the CNN model, the input contains an entire dialogue. We experiment with two embedding layers, one with SSWE embeddings, and the other with GloVe embeddings. Figure 4 presents detailed description. Gupta et al. showed that SSWE embeddings capture sentiment information and GloVe embeddings capture semantic information in the continuous representation of words. Similar to the CNN model, here also, we input the word indices of dialogue. We pad input sequences with zeros so that each sequence has length n.

The architecture consists of two LSTM layers after each embedding layer. The LSTM layer outputs a vector of shape 128×1. Further, concatenation of these output vectors results a vector of shape 256×1. In the end, we have a hidden layer followed by a softmax layer. The output from the softmax layer is a vector of shape 4×1 which refers to class probabilities for the four classes. We used Keras for this model.

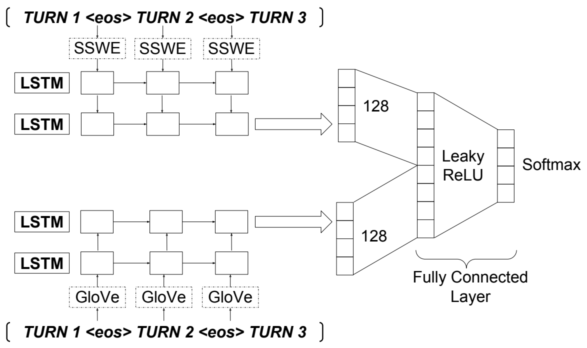


Figure 4: LSTM-I architecture.

3.2.3 Long Short-Term Memory-II (LSTM-II)

In the second approach, we use the architecture shown in Figure 5. This model consists of embedding layers, LSTM layers, a dense layer and a softmax layer. Here, the entire dialogue is not passed at once. Turn 1 is passed through an embedding layer which is followed by an LSTM layer. The output is a vector of shape 256×1. Turn 2 is also passed through an embedding layer which is followed by an LSTM layer. The output from Turn 2 is concatenated with the output of Turn 1 to form a vector of shape 512×1. The concatenated vector is passed through a dense layer which reduces the vector to 256×1. Turn 3 is passed through an embedding layer which is followed by an LSTM layer. The output from Turn 3 is concatenated with the reduced output of Turn 1 & 2, and the resultant vector has shape 512×1. The resultant vector is passed through a dense layer and then a softmax layer to find the probability distribution across different classes. We used Pytorch for this model.

The motivation of this architecture was derived from the Task's focus to identify the emotion of Turn 3. Hence, this architecture gives more weight to Turn 3 while making a prediction and conditions the result on Turn 1 & 2 by concatenating their output vectors. The concatenated vector of Turn 1 & 2 accounts for the context of the conversation.

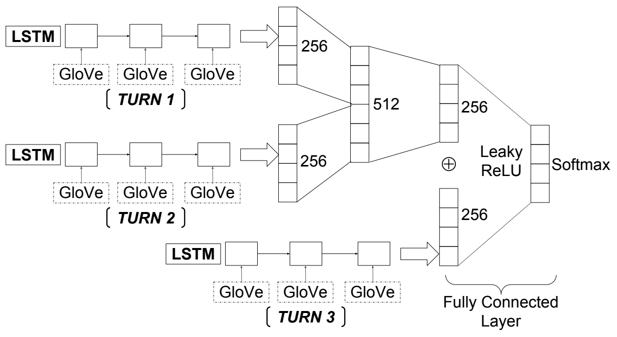


Figure 5: LSTM-II architecture.

4 Results

In Table 2, we report the performance of all the models described in the previous section. We train each model multiple (=5) times and compute the mean of scores.

Algorithm	$Score_{dev}$	$Score_{test}$
SVM	0.46	0.41
LR	0.44	0.40
SVM (tf-idf weighted averaging)	0.42	0.38
LR (tf-idf weighted averaging)	0.37	0.34
CNN	0.632	0.612
LSTM-I	0.677	**0.667**
LSTM-II	**0.684**	0.661

Table 2: Model performance on dev & test dataset.

SVM and Logistic Regression models did not yield very good results. We attribute this to the dialogue features that we use for the models. Tf-idf weighted average of GloVe vectors performed worse than the simple average of vectors. Hand-crafted features might have performed better than our current implementation. Neural network based models had very good results, CNN performed better than classical ML models but lagged behind LSTM based models. On the test set, our LSTM-I model performed slightly better than LSTM-II model.

Hyper-parameter selection for CNN was difficult, and we restricted to LSTM for the Phase 2 (i.e. test phase). We also noticed that the LSTM model was overfitting early in the training process (4-5 epochs) and that was a challenge when searching for optimal hyper-parameters. We used grid search to find the right set of hyper-parameters for our models. We grid searched over dropout (Srivastava et al., 2014), number of LSTM layers, learning rate and number of epochs. In case of the CNN model, number of filters was an extra hyper-parameter. We used Nvidia GeForce GTX 1080 for training our models.

5 Conclusion

In this paper, we experimented with multiple machine learning models. We see that LSTM and CNN models perform far better than classical ML methods. In phase-1 of the competition (dev dataset), we were able to achieve a score of 0.71, when the scoreboard leader had 0.77. But in phase-2 (test dataset), our best score was only 0.634, when the scoreboard leader had 0.79. After phase-2 ended, we experimented more with hyper-parameters and achieved an increase in scores on the test-set (mentioned in Table 2).

Full code for the paper can be found on GitHub*.

6 Future Work

Our scores on the test dataset suggest room for improvement. Now we are narrowing down to transfer learning where the starting point for our model will be a pre-trained network on a similar task. Our assumption is, this will help in better convergence on EmoContext dataset given the dataset size is not too large.

*`https://github.com/lingo-iitgn/emocontext-19`

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Stavros Giorgis, Apostolos Rousas, John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2016. aueb. twitter. sentiment at semeval-2016 task 4: A weighted ensemble of svms for twitter sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 96–99.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *CoRR*, abs/1707.06996.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th international workshop on semantic evaluation (semeval-2016)*, pages 1–18.

BARBARA ORTUTAY. 2018. Poll: Teens prefer texting over face-to-face communication.

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 D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Daily Telegraph Reporter. 2012. Texting more popular than face-to-face conversation.

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2016. Insight-1 at semeval-2016 task 4: convolutional neural networks for sentiment classification and quantification. *arXiv preprint arXiv:1609.02746*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. volume 1, pages 1555–1565.

Xinyu Wang, Chunhong Zhang, Yang Ji, Li Sun, Leijia Wu, and Zhana Bao. 2013. A depression detection model based on sentiment analysis in micro-blog social network. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 201–213. Springer.

MARK WILSON. 2016. This startup is teaching chatbots real empathy.

Vikrant Yadav. 2016. thecerealkiller at semeval-2016 task 4: Deep learning based system for classifying sentiment of tweets on two point scale. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 100–102.