Neural Feature Extraction for Contextual Emotion Detection

Elham Mohammadi, Hessam Amini and Leila Kosseim
Computational Linguistics at Concordia (CLaC) Lab
Department of Computer Science and Software Engineering
Concordia University, Montréal, Québec, Canada
first.last@concordia.ca

Abstract

This paper describes a new approach for the task of contextual emotion detection. The approach is based on a neural feature extractor, composed of a recurrent neural network with an attention mechanism, followed by a classifier, that can be neural or SVM-based. We evaluated the model with the dataset of the task 3 of SemEval 2019 (EmoContext), which includes short 3-turn conversations, tagged with 4 emotion classes. The best performing setup was achieved using ELMo word embeddings and POS tags as input, bidirectional GRU as hidden units, and an SVM as the final classifier. This configuration reached 69.93% in terms of micro-average F1 score on the main 3 emotion classes, a score that outperformed the baseline system by 11.25%.

1 Introduction

Emotions are an intricate part of human communication. Being able to interpret and react to the emotions of others allows one to better communicate. Emotion information can be extracted from a variety of physiological sources, such as electroencephalography (EEG) signals (Zhang et al., 2016), skin temperature (Li and Chen, 2006), speech signals (Trigeorgis et al., 2016) and facial expressions (Mao et al., 2015), as well as text (Yassine and Hajj, 2010).

The rise of social media and the availability of human-written online diaries, blog posts, and comments has lead to an increase in research on the automatic detection of sentiment and emotion from textual data.

Sentiment analysis and opinion mining can range from coarse-grained binary classification of texts into positive or negative classes to finer-grained classification into a variety of emotion categories, such as happy, sad, angry, and scared. Such a classification is useful in business and marketing (Medhat et al., 2014), and a variety of downstream NLP applications, such as text-to-speech, to maintain the emotion present in text, and human-computer interaction in order to take into account the emotional state of users and make responses more human-like (Hirat and Mittal, 2015).

Fine-grained emotion detection based solely on text is a challenging task. As only linguistic cues are available, facial expressions and voice features, which are known to be discriminating (Poria et al., 2016), cannot be used. In addition, several emotions can be expressed textually by the same linguistic cues, such as emotion keywords, interjections, and emojis (Liew and Turtle, 2016). Finally, many pieces of text, especially online comments, posts, and tweets are too short to allow for correct classification. These challenges highlight the importance of using contextual information in order to detect the emotion conveyed in a piece of text.

The goal of this work is to investigate the effectiveness of different models using neural networks and Support Vector Machines (SVM) for contextual emotion detection in textual conversations.

2 Related Work

Textual emotion detection has typically been addressed as a multi-class classification task, where a text is classified into different emotional categories, ranging from basic emotions to finer-grained emotional classes. Studies focusing on emotion detection have made use of different corpora and different evaluation metrics.

Proceedings of Recent Advances in Natural Language Processing, pages 785–794,
Varna, Bulgaria, Sep 2–4, 2019.
https://doi.org/10.26615/978-954-452-056-4_091

Dini and Bittar (2016) broke down the task of emotion detection from tweets into a cascade of decisions: classifying tweets into emotional and non-emotional categories, and then tagging the emotional tweets with the appropriate emotion label. For the latter, they compared a symbolic system using gazetteers, regular expressions, and graph transformation, with a machine learning system using a linear classifier with words, lemmas, noun phrases, and dependencies as features. Using their collected corpus of emotional tweets, the rule-based approach achieved an F1 score of 0.41, while the machine learning approach yielded an F1 score of 0.58 on 6 emotion classes.

Mohammad and Bravo-Marquez (2017) made use of an SVM regression model to determine the intensity of 4 emotions: anger, fear, joy, and sadness in a dataset of tweets that they have previously collected and annotated. As features, they used word and character n-grams, word embeddings trained using the word2vec skip-gram model (Mikolov et al., 2013), and affect-related lexical features. Using the Pearson correlation coefficient as evaluation metric, they demonstrated that word embeddings yield better results than n-gram features. They achieved their best average result of 0.66, using a combination of word embeddings and lexical features.

Abdul-Mageed and Ungar (2017) also collected their own dataset of emotional tweets using emotion hashtags. They trained word embeddings on the training data, employed a gated recurrent neural network (Cho et al., 2014) as a classifier and achieved an average F1 score of 0.87 over 3 emotion datasets, labelled with 8 emotions.

Abdullah and Shaikh (2018) proposed an approach to detect the intensity of affect in tweets. Their features include feature vectors extracted using the *AffectiveTweets* package of *Weka* (Holmes et al., 1994), as well as word2vec and doc2vec (Le and Mikolov, 2014) embeddings. They developed three models using different subsets of the feature set as input to either a dense feed-forward network or a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) network. Using the dataset of SemEval 2018 task 1 (Affect in Tweets) (Mohammad et al., 2018), they achieved their best Spearman correlation score of 0.69 over 4 emotions by averaging over the outputs of the three models.

More recently, Khanpour and Caragea (2018) focused on domain-specific emotion detection. They created a dataset of 2107 sentences taken from online forums on the Cancer Survivors Network website[1]. In order to combine the strengths of lexicon-based and machine learning approaches, they proposed a model that uses word2vec embeddings as input to a Convolutional Neural Network (CNN) (LeCun et al., 1999). The CNN generates feature vectors which are then augmented with domain-specific lexical features. The combined features are then used as input to an LSTM network which classifies the texts into 6 different emotion categories.

While most of the literature has focused on the detection and assessment of emotions in online textual data, few researchers have investigated emotion detection in textual conversations. We argue that the detection of emotions from dialogues poses new challenges compared to emotion detection from monologues, as the utterances made by different interlocutors can influence differently the emotional state of a speaker.

In this work, we investigate the effectiveness of neural feature extraction for the task of emotion detection in short dialogues.

3 Dataset and Task

The dataset used in this work is taken from Chatterjee et al. (2019b). It consists of short 3-turn dialogues between two speakers (turn 1 uttered by speaker 1, turn 2 uttered by speaker 2, and turn 3 uttered by speaker 1 again). Table 1 shows two samples of the dataset[2].

The goal is to detect the emotion of speaker 1 in turn 3, taking into account the previous turns. The data is annotated with 4 emotions: *happy*, *angry*, *sad*, and *others*. In order to simulate a real-life task, the distribution of the labels in the dataset is highly imbalanced: 50% of the training data belongs to the *others* class, while 14%, 18%, and 18% of the training data is dedicated to classes *happy*, *angry*, and *sad*, respectively. The test and development sets are even more imbalanced, with 85% of the samples labelled as *others*. Table 2 summarizes some statistics of the dataset.

[1] `https://csn.cancer.org/`

[2] Samples are taken from `https://competitions.codalab.org/competitions/19790`.

ID	Turn1 (Speaker1)	Turn2 (Speaker2)	Turn3 (Speaker1)	Label (of Turn3)
156	You are funny	LOL I know that. :p	☺	happy
187	Yeah exactly	Like you said, like brother like sister ;)	Not in the least	others

Table 1: Two sample dialogues from the EmoContext 2019 dataset.

Dataset	Label	# of Samples	Percentage	Average # of Tokens		
				Turn 1	Turn 2	Turn 3
Train	*happy*	4243	14%	4.873	7.195	3.825
	angry	5506	18%	5.107	6.859	5.457
	sad	5463	18%	4.608	6.450	4.829
	others	14948	50%	4.232	6.493	4.153
	All	30160	100%	4.550	6.650	4.467
Development	*happy*	142	5%	4.761	7.444	3.690
	angry	150	5%	4.647	7.347	4.867
	sad	125	5%	4.624	6.200	5.192
	others	2338	85%	4.245	6.546	4.143
	All	2755	100%	4.311	6.620	4.207
Test	*happy*	284	5%	5.063	6.845	3.493
	angry	298	5%	4.470	6.456	4.856
	sad	250	5%	5.000	6.632	4.936
	others	4677	85%	4.279	6.601	4.143
	All	5509	100%	4.362	6.607	4.184
All	*happy*	4669	12%	4.881	7.181	3.801
	angry	5954	16%	5.063	6.851	5.412
	sad	5838	15%	4.626	6.452	4.842
	others	21963	57%	4.243	6.521	4.150
	All	38424	100%	4.506	6.642	4.408

Table 2: Statistics of the EmoContext 2019 dataset.

4 The Model

Figure 1 shows the overall architecture of our model for the task of contextual emotion detection. The model is composed of two main components: 1) the neural feature extractor and 2) the classifier.

4.1 The Neural Feature Extractor

As shown in Figure 1, the neural feature extractor is a recurrent neural network with an attention mechanism. The feature extractor is responsible for creating dense vector representations for each dialogue turn. As a result, the model uses 3 feature extractors, one for each dialogue turn.

Each neural feature extractor is composed of an input layer, a recurrent layer, and an attention layer, explained below.

The Input Layer takes as input the vector representations of each word in the corresponding dialogue turn. Each dialogue turn is a sequence of tokens, represented as a vector $[x_{i,1}, x_{i,2}, \ldots, x_{i,t}, \ldots, x_{i,n}]$, where $x_{i,t}$ is the corresponding vector for the t-th word in the i-th dialogue turn, and n is the length of the i-th turn. The vector representation for each token ($x_{i,t}$) is composed of the word embedding corresponding to the token, concatenated with a one-hot representation of the token's part-of-speech (POS) tag.

The Recurrent Layer takes as input the token vectors ($[x_{i,1}, x_{i,2}, \ldots, x_{i,t}, \ldots, x_{i,n}]$), and processes them in a forward and a backward passes. In the forward pass, the content value of the hidden layer at a specific time-step is calculated using the value of the input at the current time-step, and the content value of the hidden layer in the previous time-step.

Equation 1 shows how the content value of the hidden layer is calculated at a specific time-step t, where x_t represents the input value in the current time-step, and h_t and $h_{t\pm1}$ represent the content value of the hidden node in the current and previous/next time-steps (in the forward or backward pass), respectively, and f_h is the function that calculates the value of h_t using x_t and $h_{t\pm1}$. Subsequently, the output of the hidden layer is calculated using Equation 2, where y_t is the output of the hidden layer at time-step t, and f_y is the function that calculates the output value based on h_t.

$$h_t = f_h(x_t, h_{t\pm1}) \tag{1}$$

$$y_t = f_y(h_t) \tag{2}$$

The Attention Layer is a function that automatically assigns weights to the output of the recurrent layer at each time-step, and calculates the weighted sum of the outputs using their corresponding weights (Vaswani et al., 2017). Following several works that have shown significant improvement in text classification with the use of attention (e.g. Yang et al., 2016; Zhou et al., 2016; Wang et al., 2016; Cianflone et al., 2018), we incorporated an attention mechanism in our contextual emotion detection framework. Equation 3 shows the overall mechanism of our attention layer, where $\omega_{t'}$ represents the corresponding weight for the output of the recurrent layer at time-step t' in, and n is the number of time-steps (i.e. the length of the dialogue turn).

$$Attention = \sum_{t'=1}^{n} y_{t'}\omega_{t'} \tag{3}$$

In our model, the weights are calculated by applying a single N-to-1 feed-forward layer on

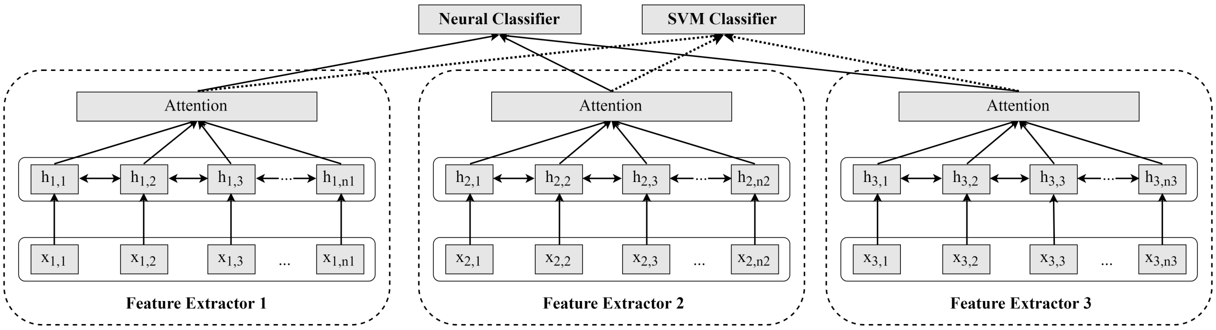


Figure 1: Architecture of the model.

the output of the recurrent layer at each time-step (where N is the size of the output of the recurrent layer), concatenating the results, and applying a softmax over them. Equations 4 and 5 show the mechanisms used to calculate the weights, where w corresponds to the weights in the single-layer neural network, and ν_t is the single value, which is the result of feeding y_t to the fully-connected layer.

$$\nu_t = y_t \times w \tag{4}$$

$$\omega = Softmax([\nu_1, \nu_2, \nu_3, \ldots, \nu_n]) \tag{5}$$

4.2 The Classifier

As shown in Figure 1, we experimented with two types of classifiers at the output layer: A fully-connected neural network, followed by a softmax activation function, and an SVM, which takes as input the neural representations generated by the 3 latent feature extractors for each dialogue turn.

The neural classifier is trained jointly with the neural feature extractors, while the SVM classifier is completely trained after each training epoch of the neural network, using the features extracted by the 3 neural feature extractors.

5 Experimental Setup

The neural network components of our model were developed using PyTorch (Paszke et al., 2017) and the SVM was developed using the Scikit-learn library (Pedregosa et al., 2011). In this section, we will explain the different setups that we experimented for the task of contextual emotion detection.

5.1 Word Embeddings

In order to test our model, we experimented with two different pretrained word embeddings. As the first word embedder, we chose GloVe (Pennington et al., 2014), which is pretrained on 840B tokens of web data from Common Crawl, and provides 300d vectors as word embeddings. As our second word embedder, we experimented with ELMo (Peters et al., 2018), which produces word embeddings of size 1024, and is pretrained on the 1 Billion Word Language Model Benchmark[3] (Chelba et al., 2014).

The main reason for choosing these two word embedders was to evaluate the effect of their embedding mechanisms for our task. As opposed to GloVe which assigns a word embedding to each token, the ELMo word embedder calculates the embedding for each token from its constituent characters by also taking into account its textual context. We suspected that this approach would lead to better results in our task (see Section 6).

5.2 POS Tags

The spaCy library[4] was used for tokenization and POS tagging, and the Penn Treebank tagset standard (Marcus et al., 1993) was followed for assigning POS tags to tokens. This lead to one-hot vectors for POS information of size 51.

5.3 Recurrent Units

Long Short-term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRU) (Cho et al., 2014) were both experimented with as the building blocks of the recurrent layer. For both LSTM and GRU, 2 layers of 25 bidirectional recurrent units were stacked.

5.4 Neural Network Optimization

The Adam optimizer (Kingma and Ba, 2014) with a learning rate of 10^{-4} was used to train the neu-

[3] The selected versions of GloVe and ELMo lead to the best results in our task. We also experimented with other versions of the two, but their performances were inferior.

[4] https://spacy.io/

ral network. Cross-entropy with class weights was used as the loss function. In order to handle the imbalanced class distribution in the dataset (see Table 2), the corresponding weight for each class was calculated proportional to the inverse of the number of samples for that class in the training data. This way, more penalty was applied to the network when an error was made on a sample from a minority class rather than a more frequent one.

Minibatches of size 32 were used during training and testing, and zero-padding was applied in order to handle different input sequence lengths. In order to minimize padding, samples with similar average lengths of tokens over the three turns were put in the same batch.

Finally, in order to avoid the exploding gradient problem (Pascanu et al., 2012), gradient clipping with a norm of 0.5 was applied.

5.5 SVM Hyperparameters

The SVM utilizes a polynomial kernel with degree of 4. To set the parameter γ, the `svm.SVC` model in Scikit-learn was initiated with its parameter `gamma` set to `auto`, which automatically sets γ to the inverse of the number of features extracted by the neural feature extractor. In our model, this value was set to 1/150, since each of the three neural feature extractors extracts 50 features from the dialogue turn that it handles.

5.6 Overall Training Process

As indicated in Section 4.2, the neural classifier was trained jointly with the neural feature extractors, while the SVM was trained separately after each epoch, using the extracted features on the training data.

The models with either neural or SVM classifier were trained for 50 epochs, and the model's parameters were saved after each training epoch. The optimal parameters were then picked as the ones that led to the highest micro-average F1 score on the three main emotion classes (all except class *other*) on the development dataset. This final model with the optimal trained parameters was then evaluated on the test set.

6 Results

The official evaluation metric used at the EmoContext shared task is the micro-average F1 score over the three main emotion classes, i.e. *happy*, *angry*, and *sad* (ignoring the 4th class, *others*).

Figure 2 shows the performance of each model on the development dataset, throughout the training process. As Figure 2 shows, using both the neural classifier and the SVM classifier, the models with GRU as the recurrent units and ELMo embeddings as input features were generally superior to the others.

The notation *<type-of-recurrent-unit>+<type-of-classifier> with <type-of-input-features>* is used in the rest of the paper, to refer to each model; for example, *LSTM+NN with GloVe* refers to the model that uses LSTM units in the recurrent layer, fully-connected neural layer as classifier, and GloVe embeddings as input; whereas *GRU+SVM with ELMo+POS* denotes the version of our model with GRU units in the recurrent layer, SVM as the classifier, and ELMo embeddings and one-hot encoded POS tags as input.

As indicated in Section 5.6, the final versions of the models were chosen based on their performance on the development dataset, i.e. for each model, the final set of trained parameters were the one that yielded the maximum micro-average F1 score on the three emotion classes on the development dataset.

The results achieved from our models are also compared with the baseline system, provided by the EmoContext 2019 shared task (Chatterjee et al., 2019a). The baseline system is composed of a neural network with 128 LSTM units in the hidden layer, and as input features, uses the GloVe word embeddings, pretrained on 6 billion tokens from Wikipedia 2014 and the Gigaword 5 corpus[5].

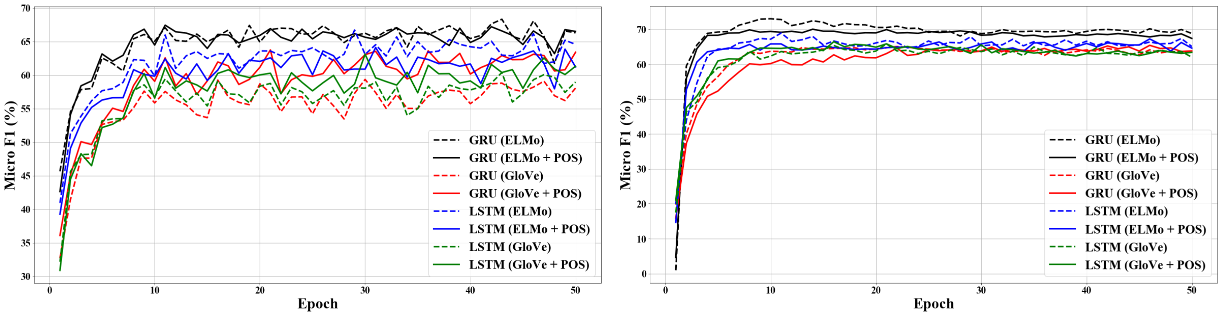
Table 3 shows the performance of each model[6], where the best micro-average F1 scores are highlighted in bold. The results show that the model *GRU+SVM with ELMo* yields the best performance of 73.03% on the development data, while the model *GRU+SVM with ELMo+POS* outperforms all the other models on the test dataset with a micro-average F1 score of 69.93%, by being marginally better than *GRU+SVM with ELMo*[7].

The results also show that, with the exception of the two models *LSTM+NN with GloVe* and *GRU+NN with GloVe* which have inferior perfor-

[5] https://catalog.ldc.upenn.edu/LDC2011T07

[6] At the time of writing this paper, the F1 score of the baseline model had not been released for each emotion class.

[7] The result that we submitted to the EmoContext shared task was slightly higher than the current result (70.72%), which was achieved by also using the features from the development dataset to train the SVM classifier.



(a) With the neural classifier.

(b) With the SVM classifier.

Figure 2: Performance of the model (in micro-average F1 score over the three main emotion classes) on the development dataset, as a function of the number of training epochs

mance than the baseline system on the test dataset, all the other models significantly outperform the baseline model on both development and test data.

7 Discussion

To better understand the results, we analyzed the effect of different components of the model.

7.1 Effect of Input Features

The results in Table 3 demonstrate that the models that use ELMo as word embeddings have a significantly higher performance than the ones with GloVe. We believe that this is due to two main reasons: 1) The ELMo word embedder is character-based, which allows it to better handle out-of-vocabulary words, and 2) ELMo takes into account the textual context of the token when extracting the word embedding.

Table 3 also shows that the use of POS tags leads to a significant improvement with the models that utilize GloVe word embeddings. On the other hand, for the models that use ELMo embeddings as input, this is not the case. As Table 3 shows, in several occasions, the use of POS tags has even reduced the performance of ELMo-based models. We believe that, in the case of GloVe, where the word embeddings are context-independent, POS tags can improve token representations to also take into account the textual context in which the tokens have occurred. However, since ELMo already takes into account the textual context when extracting the token representations, POS tags do not help much and can even be redundant in some cases.

7.2 Effect of the Recurrent Units

The results in Table 3 show that for the models that incorporate ELMo embeddings, the ones that use GRU in their recurrent layer significantly outperform the ones with LSTM; however, this behavior cannot be observed in GloVe-based models, as we can see several cases, where the LSTM-based models are slightly better.

It could be concluded that, for the current task, since GloVe word embeddings are context-independent, a stronger recurrent unit is required to capture the context, while in the case of ELMo, where context is already taken into account, a simpler recurrent unit such as GRU is enough while being less prone to overfitting.

7.3 Effect of the Classifiers

Table 3 shows that, in almost all cases, the models with the SVM classifier significantly outperform the ones with the neural classifier. We believe that, although the neural network may have been able to reach similar results as the SVM, the latter reached this performance using less fine-tuning due to its explicit design to optimize the margin size between classes.

On another note, Figure 2 shows that the models with the SVM classifier were significantly more robust and demonstrated much less performance fluctuation during training than the ones with the neural classifier. We believe that this is due to the more deterministic nature of SVM in comparison to the neural networks.

However, the most important drawback of the SVM was the training time: since the SVM classifier was trained separately from feature extractors, training it entailed additional training time to the model. All being said, we believe that, if training

Model	Input Feature	F1 Score on Development Data				F1 Score on Test Data			
		happy	*angry*	*sad*	**Micro**	*happy*	*angry*	*sad*	**Micro**
BASELINE		–	–	–	58.61	–	–	–	58.68
LSTM+NN	GloVe	53.45	67.37	59.39	60.40	52.05	67.02	52.89	57.60
	Glove+POS	58.33	68.28	62.54	63.15	58.03	68.68	59.77	62.35
	ELMo	63.88	67.61	69.14	66.74	62.07	65.14	67.37	64.74
	ELMo+POS	59.03	65.92	68.42	64.13	63.95	62.06	68.56	64.54
GRU+NN	GloVe	47.79	65.67	66.19	59.36	51.34	65.80	58.04	58.26
	Glove+POS	57.94	68.39	64.74	63.77	61.24	69.27	57.82	62.99
	ELMo	64.26	67.58	74.71	68.34	63.32	66.21	69.73	66.33
	ELMo+POS	65.48	65.15	73.12	67.46	62.20	66.40	71.64	66.53
LSTM+SVM	GloVe	54.85	67.06	66.91	65.83	53.00	68.05	57.30	62.84
	Glove+POS	61.30	68.95	66.20	66.46	60.30	67.32	60.84	63.26
	ELMo	65.64	65.91	73.44	68.94	63.78	65.25	70.10	66.45
	ELMo+POS	62.31	68.44	68.73	67.25	63.37	64.09	68.86	67.28
GRU+SVM	GloVe	50.21	68.31	71.00	65.08	50.22	70.26	60.13	62.02
	Glove+POS	52.77	68.73	66.67	65.42	56.00	69.68	57.96	63.11
	ELMo	67.33	67.83	75.40	**73.03**	65.08	68.83	73.07	69.39
	ELMo+POS	68.21	66.26	74.46	70.03	64.71	69.13	71.05	**69.93**
AVERAGE		59.55	67.34	68.82	65.96	59.42	67.07	64.07	64.22

Table 3: The performance of each model on the development and test datasets, in terms of F1 score on each emotion class, and micro-average F1 over the three main emotion classes. The AVERAGE is computed over the proposed models and does not include the baseline.

time is not a concern, the SVM classifier is a better option than the neural one.

An interesting finding regarding the SVM classifier is that, in contrast to the neural network where applying class-weights to the loss function helped improve the performance of the models, applying class-weights to the SVM decreased its performance.

7.4 A Closer Look at Emotion Classes

The row labelled *AVERAGE* in Table 3 provide information regarding the difficulty of detecting each class. Table 3 shows that, among the three main classes, *happy*, *angry*, and *sad*, the class *happy* was the most difficult to detect.

Table 2 shows that the low average F1 score for class *happy* is probably due to the significantly smaller number of samples with this class in the training data (14%) in comparison to the samples from the other two emotion classes (18% and 18%). Although the weighted loss functions (see Section 5.4) somehow managed to handle the imbalanced class distribution in the data, the optimal weights are not necessarily proportional to the inverse of the frequency of classes.

7.5 Quality of the Extracted Neural Features

To better understand the contribution of the extracted neural features from the feature extractors, we calculated the mutual information between the values of each neural feature and the classes.

Figure 3 shows the average and the standard deviation of the mutual information between the features extracted from each neural feature extractor in each model and the classes in the training data. Since both the neural and the SVM classifiers use the same set of neural features, we did not differentiate between models with similar neural feature extractors and different classifiers.

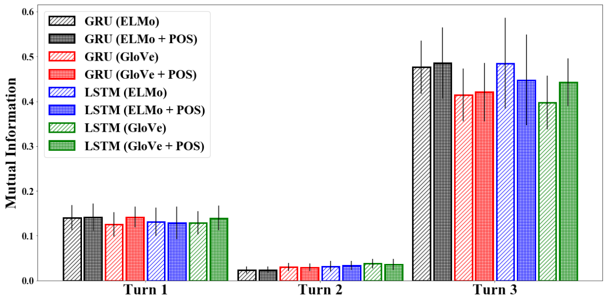


Figure 3: The average mutual information between the features of each dialogue turn, extracted by each neural feature extractor, and the classes.

As Figure 3 shows, in all cases, the features extracted from the third turn of the conversation have the highest mutual information with the classes, and the ones from the second turn have the lowest. This agrees with the nature of the dataset, where the label is assigned to the emotion of speaker 1 (who uttered dialogue turns 1 and 3) after the third turn is uttered. This also indicates that the nature of emotion detection in the context of dialogues is different from that in monologues in two ways: not only do the utterances by different speakers contribute differently to the emotional state of a speaker, but also the timing of the utterances by the same speaker has an impact on the contribution of that utterance to the emotion classification.

Figure 3 shows that the difference in average mutual information between the extracted features and the emotional classes are higher for features from the third (i.e. the most recent) dialogue turn. As expected, the features from the ELMo-based models have significantly higher mutual information with the classes than the ones from the GloVe-based models. The features from the third dialogue turn in models with GRU have slightly higher mutual information than the ones from the models with LSTM. However, the standard deviation between the features extracted by the GRU-based models are significantly smaller than the ones with LSTM, showing that between the models with LSTM and the ones with GRU, the features extracted from the models with GRU had more similar amount of contribution to the classification task than the ones from the LSTM. This has led to the GRU-and-ELMo-based models outperforming the others.

A surprising finding is that the neural features extracted by the GRU-based models from the second dialogue turn have the least mutual information with the classes in comparison to the ones from the other models. Observing this, we experimented with our classifiers by disregarding the neural features from the second turn; however, this lead to a slight performance drop. We hypothesize that, although the neural features from the second dialogue turns bring only a small contribution to the classification, the GRU-based models tend to focus more on the features from the other two turns. This leads to the second feature extractor being less focused upon, and as a result, being less trained than the ones from other models.

8 Conclusion and Future Work

In this paper, we proposed a model for the task of contextual emotion detection. We evaluated our model with the EmoContext 2019 shared task dataset (Chatterjee et al., 2019b), which consists of 3 turn conversations tagged with one of the labels: *happy*, *sad*, *angry*, and *others*, based on the emotion present in the last dialogue turn.

The proposed model utilizes an attention-based recurrent neural network. We experimented with GloVe and ELMo embeddings, alongside POS tags as input, LSTM and GRU as recurrent units, and a neural or an SVM classifier. The best result on the test dataset was achieved with ELMo and POS tags as input, GRU as recurrent units, and SVM as the final classifier. Using this setup, we reached a performance of 69.93% in terms of micro-average F1 score which is a significant improvement over the baseline of 58.68%.

Three future directions can be proposed. The first is to investigate a more effective way of handling the imbalanced distribution of labels in the dataset. As a example, methods for finding the optimal class-weights for training the models can be investigated.

Secondly, the use of different number of features for each dialogue turn can be studied. As shown in Figure 3, features extracted from different dialogue turns had different levels of contribution to the final classification. In that case, more features could be extracted from turns 3 and 1 (uttered by the same speaker) in comparison to turn 2, which has the least contribution to the classification.

Lastly, knowing that the SVM classifier is capable of outperforming the neural one, studies can be performed in order to make the extracted features more suitable for the SVM classifier.

Acknowledgments

We would like to express our gratitude to Parsa Bagherzadeh for insightful discussions throughout the course of this work. We would also like to thank the anonymous reviewers for their comments on an earlier version of this paper.

This work was financially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

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 (ACL 2017)*. Vancouver, Canada, pages 718–728.

Malak Abdullah and Samira Shaikh. 2018. Teamuncc at SemEval-2018 task 1: Emotion detection in English and Arabic tweets using deep learning. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval 2018)*. New Orleans, Louisiana, USA, pages 350–357.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior* 93:309–317.

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

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *15th Annual Conference of the International Speech Communication Association (INTERSPEECH 2014)*. Singapore.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. Doha, Qatar, pages 1724–1734.

Andre Cianflone, Yulan Feng, Jad Kabbara, and Jackie Chi Kit Cheung. 2018. Let's do it "again": A first computational approach to detecting adverbial presupposition triggers. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*. Melbourne, Australia, pages 2747–2755.

Luca Dini and André Bittar. 2016. Emotion analysis on twitter: The hidden challenge. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Portorož, Slovenia.

Ruchi Hirat and Namita Mittal. 2015. A survey on emotion detection techniques using text in blogposts. *International Bulletin of Mathematical Research* 2(1):180–187.

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

Geoffrey Holmes, Andrew Donkin, and Ian H. Witten. 1994. Weka: a machine learning workbench. In *Proceedings of ANZIIS 1994 - Australian New Zealnd Intelligent Information Systems Conference*. Brisbane, Australia, pages 357–361.

Hamed Khanpour and Cornelia Caragea. 2018. Fine-grained emotion detection in health-related online posts. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*. Brussels, Belgium, pages 1160–1166.

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

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*. Bejing, China, pages 1188–1196.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. 1999. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, Springer, pages 319–345.

Lan Li and Ji-hua Chen. 2006. Emotion recognition using physiological signals. In *Proceedings of 16th International Conference on Artificial Reality and Telexistence (ICAT 2006)*. Hangzhou, China, pages 437–446.

Jasy Suet Yan Liew and Howard R. Turtle. 2016. Exploring fine-grained emotion detection in tweets. In *Proceedings of the NAACL Student Research Workshop*. San Diego, California, USA, pages 73–80.

Qi-rong Mao, Xin-yu Pan, Yong-zhao Zhan, and Xiang-jun Shen. 2015. Using kinect for real-time emotion recognition via facial expressions. *Frontiers of Information Technology & Electronic Engineering* 16(4):272–282.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

Walaa Medhat, Ahmed Hassan, and Hoda Korashy. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal* 5(4):1093–1113.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Workshop Proceedings of the International Conference on Learning Representations (ICLR 2013)*. Scottsdale, Arizona, USA.

Saif Mohammad and Felipe Bravo-Marquez. 2017. Emotion intensities in tweets. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*. Vancouver, Canada, pages 65–77.

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 (SemEval 2018)*. Association for Computational Linguistics, New Orleans, Louisiana, pages 1–17.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *Computing Research Repository* arXiv:1211.5063v1.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS 2017 Autodiff Workshop*. Long Beach, California, USA.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12(Oct):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 2014)*. Doha, Qatar, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018)*. New Orleans, Louisiana, USA, pages 2227–2237.

Soujanya Poria, Erik Cambria, Newton Howard, Guang-Bin Huang, and Amir Hussain. 2016. Fusing audio, visual and textual clues for sentiment analysis from multimodal content. *Neurocomputing* 174(PA):50–59.

George Trigeorgis, Fabien Ringeval, Raymond Brueckner, Erik Marchi, Mihalis A Nicolaou, Björn Schuller, and Stefanos Zafeiriou. 2016. Adieu features? End-to-end speech emotion recognition using a deep convolutional recurrent network. In *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2016)*. Shanghai, China, pages 5200–5204.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, Curran Associates, Inc., Long Beach, California, USA, pages 5998–6008.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*. Austin, Texas, USA, pages 606–615.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*. San Diego, California, USA, pages 1480–1489.

Mohamed Yassine and Hazem Hajj. 2010. A framework for emotion mining from text in online social networks. In *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*. Sydney, Australia, pages 1136–1142.

Yong Zhang, Xiaomin Ji, and Suhua Zhang. 2016. An approach to EEG-based emotion recognition using combined feature extraction method. *Neuroscience Letters* 633:152–157.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*. Berlin, Germany, pages 207–212.