Automatic Detection of Emotions and Distress in Textual Data

Elham Mohammadi

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science at

Concordia University

Montréal, Québec, Canada

October 2019

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Elham Mohammadi**

Entitled: **Automatic Detection of Emotions and Distress in Textual Data**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Aiman Hanna

_____ Examiner
Dr. Tristan Glatard

_____ Examiner
Dr. Olga Ormandjieva

_____ Supervisor
Dr. Leila Kosseim

Approved by _____
Dr. Lata Narayanan, Chair
Department of Computer Science and Software Engineering

_____ 2019

_____
Dr. Amir Asif, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Automatic Detection of Emotions
and Distress in Textual Data

Elham Mohammadi

Online data can be analyzed for many purposes, including the prediction of stock market, business, and political planning. Online data can also be used to develop systems for the automatic emotion detection and mental health assessment of users. These systems can be used as complementary measures in monitoring online forums by detecting users who are in need of attention.

In this thesis, we first present a new approach for contextual emotion detection, i.e. emotion detection in short conversations. The approach is based on a neural feature extractor, composed of a recurrent neural network with an attention mechanism, followed by a final classifier, that can be neural or SVM-based. The results from our experiments showed that, by providing a higher and more robust performance, SVM can act as a better final classifier in comparison to a feed-forward neural network.

We then extended our model for emotion detection, and created an ensemble approach for the task of distress detection from online data. This extended approach utilizes several attention-based neural sub-models to extract features and predict class probabilities, which are later used as input features to a Support Vector Machine (SVM) making the final classification. Our experiments show that using an ensemble approach which makes use different sub-models accessing diverse sources of information can improve classification in the absence of a large annotated dataset.

The extended model was evaluated on two shared tasks, CLPsych and eRisk 2019, which aim at suicide risk assessment, and early risk detection of anorexia, respectively. The model ranked first in tasks A and C of CLPsych 2019 (with macro-average F1 scores of 0.481 and 0.268, respectively), and ranked first in the first task of eRisk 2019 in terms of F1 and latency-weighted F1 scores (0.71 and 0.69, respectively).

Acknowledgments

I would first like to express my deepest gratitude to Dr. Leila Kosseim. Three years ago, she gave me a wonderful chance to be part of her research team at CLaC lab, Concordia, and that has so far been my greatest life-changing event. Working under the supervision of Dr. Kosseim, I had the privilege to benefit from her knowledge, constant guidance, and most important of all, her passion for natural language processing, without which this thesis would not have been possible.

I would like to thank all my past and present colleagues at CLaC, who taught me many things and also made the past two years lots of fun.

I would also like to take this opportunity to thank my amazing parents, who have always supported me no matter the cost, even when I decided to move ten thousand kilometers away.

Lastly, I dedicate this thesis to my greatest inspiration: my spouse, Hessam Amini. He relentlessly supported me in making the big shift from linguistics to computer science and made me fall in love with natural language processing. Being Hessam's colleague and classmate was the best experience of my life. I will cherish every minute we spent attending different courses, studying for final exams, and working on our papers together. Our fascinating discussions regarding different technical topics have shaped a great part of my master's journey for which I will be forever grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In the past few years, social media has been commonly used as a means to express ideas, thoughts, and emotions. 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.

Online data can be analyzed for many purposes, including marketing and business (Jacobs et al., 2018; Mansar and Ferradans, 2018), stock market prediction, and political strategy planning (Giatsoglou et al., 2017). Extracting public opinion on products and services posted online can help businesses move towards what people want, leading to more profits.

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 Natural Language Processing (NLP) applications, such as text-to-speech, to maintain the emotion present in a 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).

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

The analysis of online data can also lead to an enhanced awareness of emergencies (Yin et al., 2015). Through the detection of toxicity, aggressive behavior, hate speech, and cyber bullying in online platforms, timely interventions in violent situations can be facilitated (Davidson et al., 2017; Thompson et al., 2017).

In healthcare applications, online posts have been used for detecting disease outbreaks (Ofoghi et al., 2016), finding smoking patterns (Struik and Baskerville, 2014), and the identification of adverse drug reactions (Yang et al., 2012). Another useful application is the automatic detection of mental health issues, a relatively recent field which has attracted the attention of many researchers in NLP. Corpora from Twitter, Facebook, blogs and online forums, and Reddit are used as resources to detect various mental health problems, such as anxiety, depression, suicide ideation, and eating disorders (Calvo et al., 2017).

Although automatically monitoring online forums to detect cases of mental health issues is beneficial, the elapsed time between the first signs of a mental issue and the actual detection of a potential victim can play a crucial role. Earlier detection of a harmful behavior can help moderators better handle the situation. However, to the best of our knowledge, not much research has specifically addressed the task of early detection of mental health issues. As a result, part of this thesis is dedicated to addressing the timing of risk detection as well as the detection itself.

1.2 Goal of the Thesis

The focus of this thesis is **to develop an automatic approach that can be used for the detection of emotions and different types of distress in textual data**. We focus on 3 aspects: contextual emotion detection, suicide risk assessment, and early detection of anorexia.

Deep learning approaches have significantly improved the state-of-the-art in many tasks. However, in order to achieve such results, they often rely on large annotated datasets which are in many

cases difficult to obtain. This thesis explores the use of an ensemble architecture that uses diverse sources of information, as a substitute to the use of a large body of annotated samples.

For the purpose of model generalizability, the use of handcrafted features is minimized. On the other hand, deep neural architectures are an essential part of our experiments as they can automatically extract features. We investigated the effectiveness of our model by participating to and analyzing our results in 3 international shared tasks: EmoContext 2019 (Chatterjee et al., 2019b), CLPsych 2019 (Zirikly et al., 2019), and the first task of eRisk 2019 (Losada et al., 2019).

This thesis aims to investigate the use of developments in the field of deep learning and benefit from the strengths of these methods by applying them to a use case in natural language processing in emotion detection and clinical psychology. As automatic assessment tools can be helpful as complementary measures to monitor the emotional state and mental health of online users and as this field is attracting more and more interest from the research community, this thesis can serve as a stepping stone for much more research in this area.

1.3 Contributions

This thesis makes the following contributions:

- The implementation of different deep learning models for the task of contextual emotion detection, its evaluation through the participation to the EmoContext shared task at SemEval 2019 (Mohammadi et al., 2019a), and a performance analysis of the emotion detection models and the different components used (Mohammadi et al., 2019d) (see Chapter 3).

- The implementation of an ensemble model including several attention-based deep models for the task of suicide risk assessment and achieving state-of-the-art results in tasks A and C of the CLPsych 2019 shared task (Mohammadi et al., 2019b) (see Chapter 4).

- The evaluation of the ensemble model for the task of early risk detection of anorexia and achieving state-of-the-art results in the first task of eRisk 2019 (Mohammadi et al., 2019c) (see Chapter 5).

1.4 Thesis Structure

This chapter served as an introduction to emotion detection and mental health assessment in textual data, and our motivation to apply natural language processing techniques to address these tasks. Chapter 2 presents an overview of recent approaches in natural language classification problems and common evaluation metrics. Chapter 3 gives an account of our initial classification system which is developed for the task of contextual emotion detection. Chapter 4 presents the development and evaluation of a more refined ensemble approach, aimed at suicide risk assessment. Chapter 5 discusses the results of using the ensemble approach developed in Chapter 4 to perform the task of early risk detection. Finally, Chapter 6 concludes this thesis by presenting a summary of our findings and discussing possible future directions.

Chapter 2

Background

Text classification, whose goal is to assign a piece of text to one or multiple classes, is an essential part of many Natural Language Processing (NLP) applications. In this chapter, we present previous work on textual classification for emotion detection and for mental health assessment.

Over the years, a variety of techniques have been used in text classification systems. With the availability of relatively larger corpora of annotated data, and thanks to recent advances in the field of deep learning for natural language processing, neural network architectures have gained much interest for the task of text classification (Amini et al., 2019).

Section 2.2 describes different neural network architectures that are commonly used for text classification. In Section 2.3, common metrics used to evaluate classification systems are presented. Sections 2.2 and 2.3 contain the necessary background in order to better appreciate Section 2.4, which reviews previous work that has been done in the area of emotion detection and mental health assessment in textual data.

2.1 Classical Machine Learning Approaches

Classical machine learning approaches such as Naïve Bayes classifier (Kim et al., 2006; Frank and Bouckaert, 2006), Logistic Regression (Cox, 2018; Genkin et al., 2007), K-nearest Neighbors (Jiang et al., 2012), Decision Trees (Apté et al., 1994), Random Forests (Breiman, 2001; Xu et al., 2012), and Support Vector Machines (SVMs) (Cortes and Vapnik, 1995; Manevitz and Yousef,

2001) have been commonly used in the literature for text classification. Although these methods can extract classification rules automatically, they still require manually determined discriminative linguistic features as their input (Amini et al., 2019).

Among the methods mentioned above, we will briefly explain SVMs due to the fact that they have been used and repeatedly referred to, in this thesis.

Support Vector Machine (SVM) classifiers aim to find a separative hyperplane between two classes. During the training, an SVM tries to find the separative hyperplane (a.k.a. decision boundary) that is the furthest from the support vectors. In each class, the support vectors are made up of samples that are closest to their counterparts in the other class. By using the support vectors to find the decision boundary, a new concept called *decision margin* is introduced as the distance between the decision boundary and the support vectors closest to it.

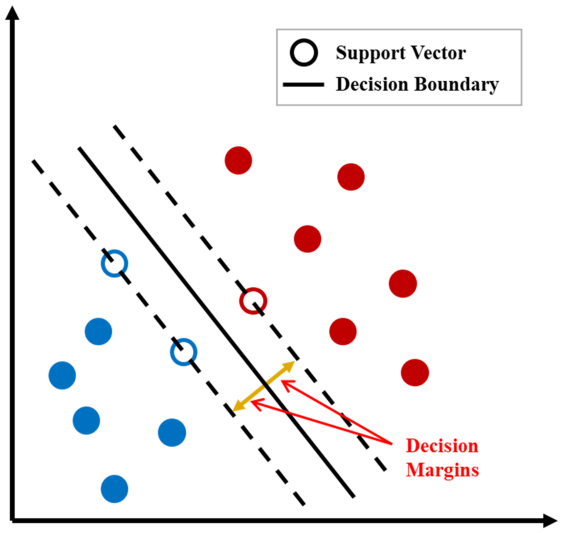

Figure 2.1: General mechanism of an SVM.

Figure 2.1 shows the support vectors, the decision boundary, and the decision margins in a classification where classes are linearly separable. As the classes are not always linearly separable, SVMs use different kernels (e.g. polynomial or sigmoid) to map the features to higher dimensions where the two classes can be separated by a hyperplane. Figure 2.2 shows an example of kernel used in an SVM.

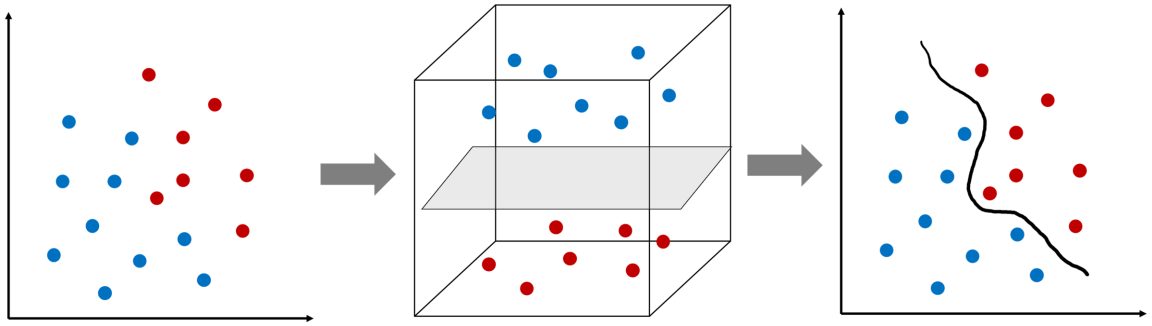

Figure 2.2: An example of kernel use in an SVM, including the mapping of features to higher dimensions, and the final decision boundary in the original problem space.

2.2 Deep Learning Approaches

Deep learning algorithms have gained much popularity in the past 10 years, due to their success in improving the state-of-the-art in many fields including NLP (LeCun et al., 2015). They have been shown capable of modeling complex patterns in large datasets by using the backpropagation algorithm which lets the system automatically find both rules and features needed for classification.

In the next section, some of the most popular neural network architectures for classification are presented.

2.2.1 Neural Network Architectures

Feed-forward Neural Networks

Feedforward neural networks are composed of one or multiple fully-connected layers, with each node in a fully-connected layer having a one-way connection to all neurons in the previous layer.

Figure 2.3 shows the architecture of a fully-connected layer and how it is connected to the previous layer.

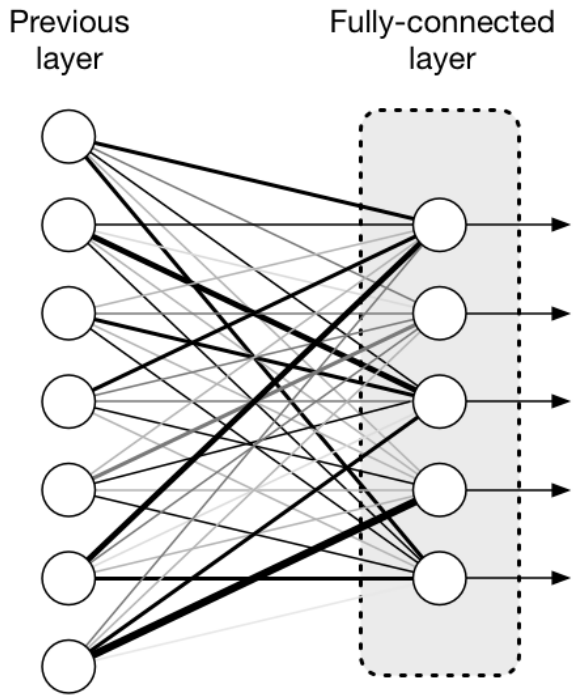

Figure 2.3: Architecture of a fully-connected layer in a neural network[1].

Assuming that a fully-connected layer of m nodes is followed by another layer of size n, the values of the fully-connected layer is calculated using Equation 1. In Equation 1, f represents a vector of the m output values of the fully-connected layer, p represents the output vector from the previous layer, and W is the matrix representing the connecting weights between the two layers, which has the size $m \times n$.

$$f = pW \tag{1}$$

Feed-forward architectures have two main drawbacks, making them undesirable to be used alone in text classification tasks. Firstly, they require fixed input sizes. This leads to problems in handling textual samples which often have different lengths. Secondly, with longer samples, the number of parameters in the network increases and this can cause overfitting.

As a result of these disadvantages, the use of fully-connected architectures is often limited to the

[1]The figure was taken from https://machinethink.net/blog/mps-matrix-multiplication/.

last layers of the neural networks that are used for text classification. To avoid the problems mentioned above, two other architectures are often used: convolutional and recurrent neural networks.

Convolutional Neural Networks

Convolutional neural networks (CNN) (LeCun et al., 1999) have been commonly used in image processing, as they take advantage of the spatial structure of images. Working under the assumption that pixels in close proximity to each other are related, CNNs use feature maps to connect patches of input to one neuron, an operation called *convolution* that allows for a smaller number of weights that are shared between the input patches and allowing for a better capturing of patterns.

But image processing is not the only field that can benefit from this architecture. CNNs have also become popular in natural language processing, due to their computational efficiency and capacity to capture generalizations. Applying the convolution concept to natural language, we can assume that the patches of input in texts are in fact N-grams (i.e. N consecutive words or characters). Figure 2.4 shows the convolution operation on textual data.

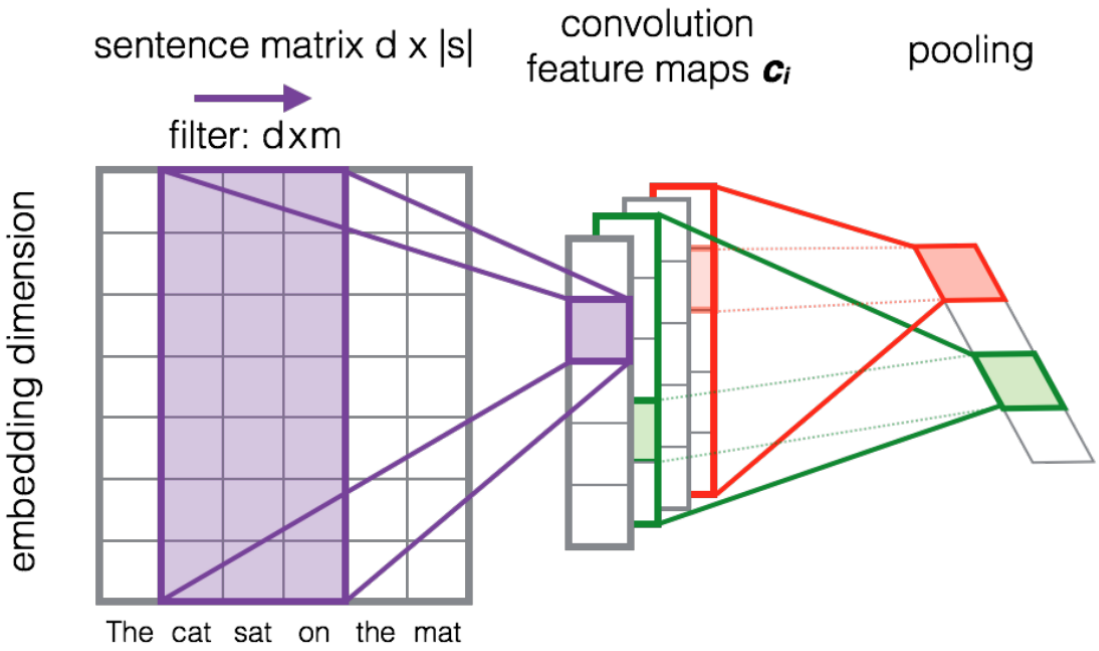

Figure 2.4: The visualization of a convolution operation in text[2].

[2]The figure was taken from http://ruder.io/text-classification-tensorflow-estimators/.

CNNs handle both problems of fully-connected feedforward networks (see Section 2.2.1) that are overfitting and handling different input sizes. However, they are unable to capture sequential information which is usually of importance in NLP tasks. That brings us to another architecture: recurrent neural networks.

Recurrent Neural Networks

A recurrent neural network (RNN) is composed of recurrent layers, that include intra-layer connections in addition to being connected to their previous and next layers.

In general, RNNs are capable of processing time-series data, to which natural language texts belong to. RNNs can be used in forward and backward passes. A forward pass is when a time-series data is fed to the RNN from the first to the last element, while in a backward pass, the data is fed from the last to the first. Figure 2.5 visualizes the unrolling of an RNN through time in a forward pass.

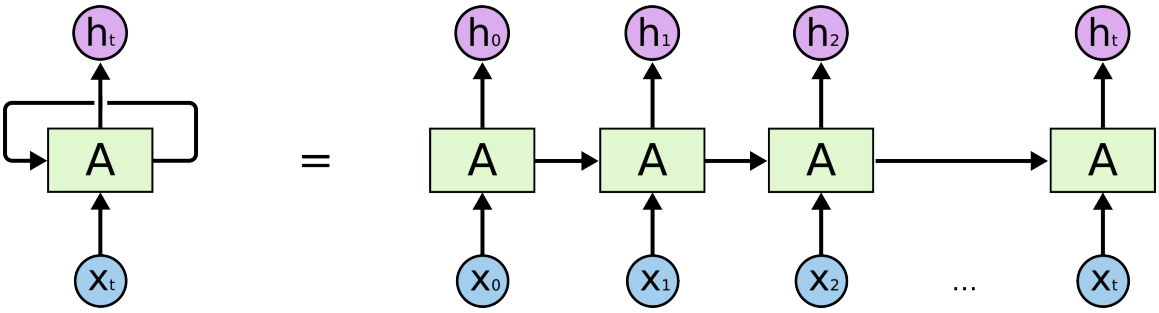

Figure 2.5: The visualization of an RNN when unrolled through time[3].

A vanilla RNN is one of the simplest RNN architectures that have been proposed. Figure 2.6 shows the architecture of a vanilla RNN, and its connection to the previous and next time-steps when used in a forward pass. The output of a vanilla RNN at time-step t (referred to as h_t in Figures 2.5 and 2.6) is calculated by Equation 2, where x_t represents the input at time-step t, $h_{t\pm1}$ refers to the output of the vanilla RNN at the previous/next time-steps (based on being in the forward/backward pass), and U represents the weights connecting the input to the RNN unit, and W stands for the weights that connect the RNN unit to itself.

[3]The figure was taken from https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

$$h_t = \tanh(x_t U + h_{t\pm1} W) \tag{2}$$

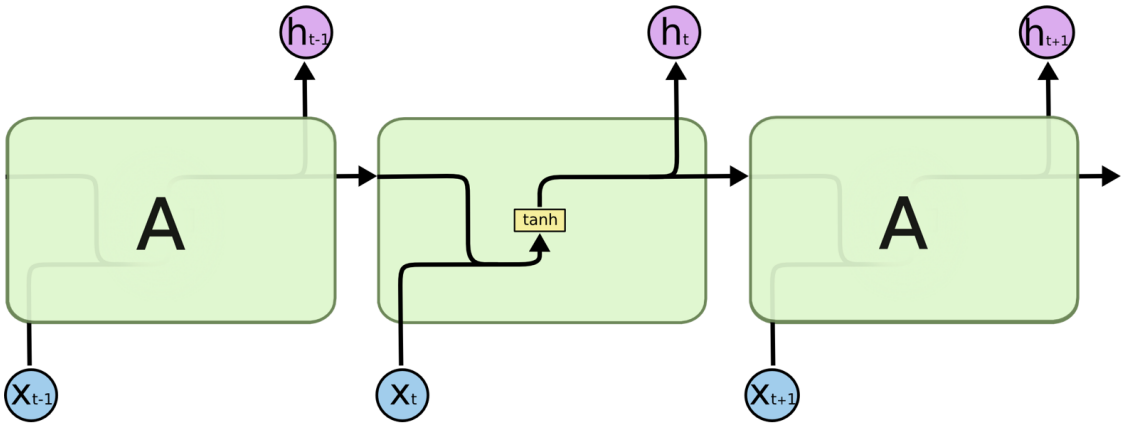

Figure 2.6: Architecture of a vanilla RNN[4].

During training, vanilla RNNs suffer from the so-called vanishing and exploding gradient problems (Bengio et al., 1994; Hochreiter and Schmidhuber, 1997), which prevents the network from learning, as the sequence of words increases. To avoid these problems, two alternate models have been proposed: 1) Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), and 2) Gated Recurrent Unit (GRU) (Cho et al., 2014).

Long Short-term Memory

Hochreiter and Schmidhuber (1997) introduced a gated recurrent architecture called the Long Short-Term Memory (LSTM), by adding *input*, *forget*, and *output* gates to the vanilla RNN. Figure 2.7 shows the architecture of an LSTM, when being used in a forward pass.

[4]The figure was taken from https://github.com/roomylee/rnn-text-classification-tf.

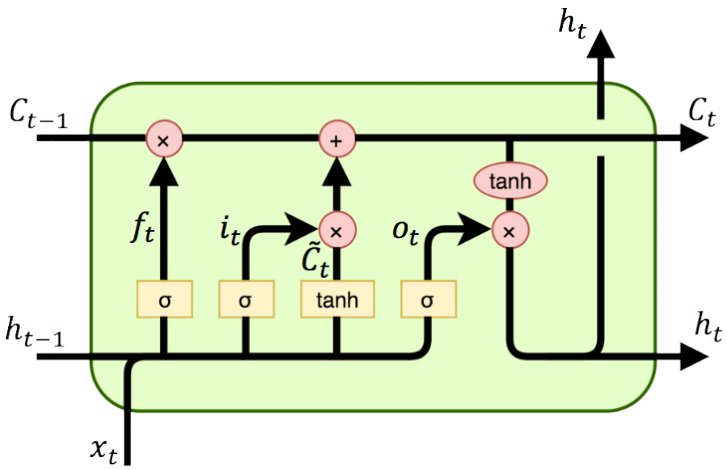

Figure 2.7: Architecture of an LSTM[5].

In an LSTM, the values of the input, forget, and output gates at a specific time-step t (referred to as i_t, f_t, and o_t) are calculated using Equations 3, 4, and 5, respectively:

$$i_t = \sigma(x_t U^i + h_{t\pm1} W^i) \tag{3}$$

$$f_t = \sigma(x_t U^f + h_{t\pm1} W^f) \tag{4}$$

$$o_t = \sigma(x_t U^o + h_{t\pm1} W^o) \tag{5}$$

where i_t, f_t, and o_t refer to the values of the input, forget, and output gates at time-step t, respectively, x_t represents the input at time-step t, $h_{t\pm1}$ represent the output of the LSTM unit at time-step t, the Us refer to the weights connecting the input to the gates, the Ws represent the connecting weights between the output of LSTM to its gates, and σ refers to the sigmoid activation function.

An LSTM includes a cell state (the memory), whose value is calculated using Equations 6 and 7.

$$\tilde{C}_t = \tanh(x_t U^g + h_{t\pm1} W^g) \tag{6}$$

$$C_t = \sigma(f_t * C_{t\pm1} + i_t * \tilde{C}_t) \tag{7}$$

Finally, Equation 8 shows how the output of an LSTM unit is calculated.

[5]The figure has been taken from https://github.com/roomylee/rnn-text-classification-tf.

$$h_t = \tanh(C_t) * o_t \tag{8}$$

LSTMs' main strength lies in handling longer sequences of input. Having the addition of input, forget, and output gates has made them less prone to the vanishing and exploding gradient problems. However, the high number of parameters (that are weights) makes them more prone to overfitting, in comparison to vanilla RNNs.

Gated Recurrent Units

More recently, a new recurrent architecture, called Gated Recurrent Unit (GRU), was presented by Cho et al. (2014). This new architecture introduced two new types of gates to the vanilla RNN, the *update* and *reset* gates. In comparison to LSTM, GRU has a relatively smaller number of parameters which makes it less prone to overfitting (Chung et al., 2014). Figure 2.8 shows the architecture of a GRU when used in a forward pass.

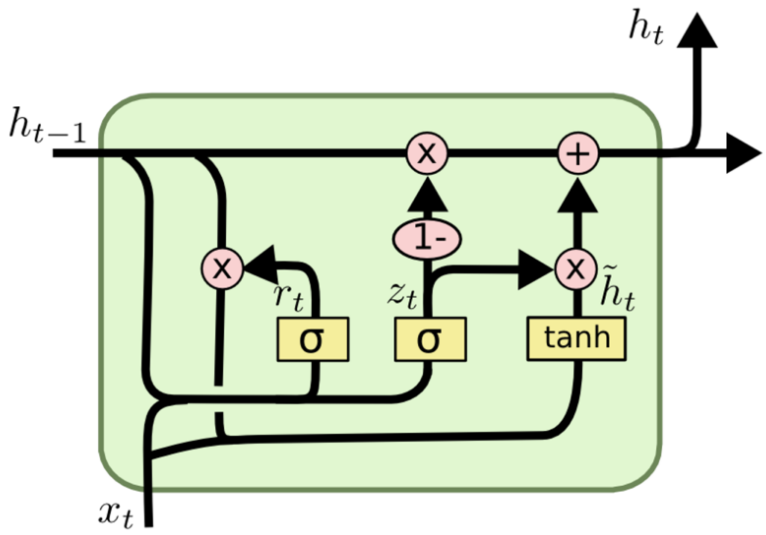

Figure 2.8: Architecture of a GRU[6].

In a GRU, the values of the update and reset gates at a specific time-step t (referred to as z_t and r_t, respectively) are calculated using Equations 9 and 10.

[6]The figure was taken from https://github.com/roomylee/rnn-text-classification-tf.

$$z_t = \sigma(x_t U^z + h_{t\pm1} W^z) \tag{9}$$

$$r_t = \sigma(x_t U^r + h_{t\pm1} W^r) \tag{10}$$

Equations 11 shows how the internal memory $\tilde{h}_t$ is updated.

$$\tilde{h}_t = \tanh(x_t U^h + (r_t * h_{t\pm1} W^h)) \tag{11}$$

And finally, Equation 12 shows how the output of a GRU is calculated.

$$h_t = (1 - z_t) * h_{t\pm1} + z_t * \tilde{h}_t \tag{12}$$

GRUs have a significantly smaller number of parameters in comparison to LSTMs and as a result, they are less prone to overfitting (Chung et al., 2014). However, they are not as powerful as LSTMs in modelling long sequences. This makes them a more suitable candidate when data is comprised of short sequences such as short conversational turns.

2.2.2 Word Embeddings

Traditionally, when feeding words to a neural network, they were represented by one-hot vectors (i.e. a vector of all zeros, except for an element having the value 1), which indicate the index of a word in a vocabulary. Embeddings are considered as matrices that are used to map these one-hot representations to a dense space with the aim of capturing semantic information about the word and also reducing sparsity (Amini et al., 2019).

The concept of word embeddings was introduced first by Bengio et al. (2003) who jointly trained word embeddings with the neural model itself in order to perform the task of language modeling which is predicting a word given its previous words.

Figure 2.9 shows the architecture of the neural word embedder of Bengio et al. (2003), where C is considered as the embedding matrix.

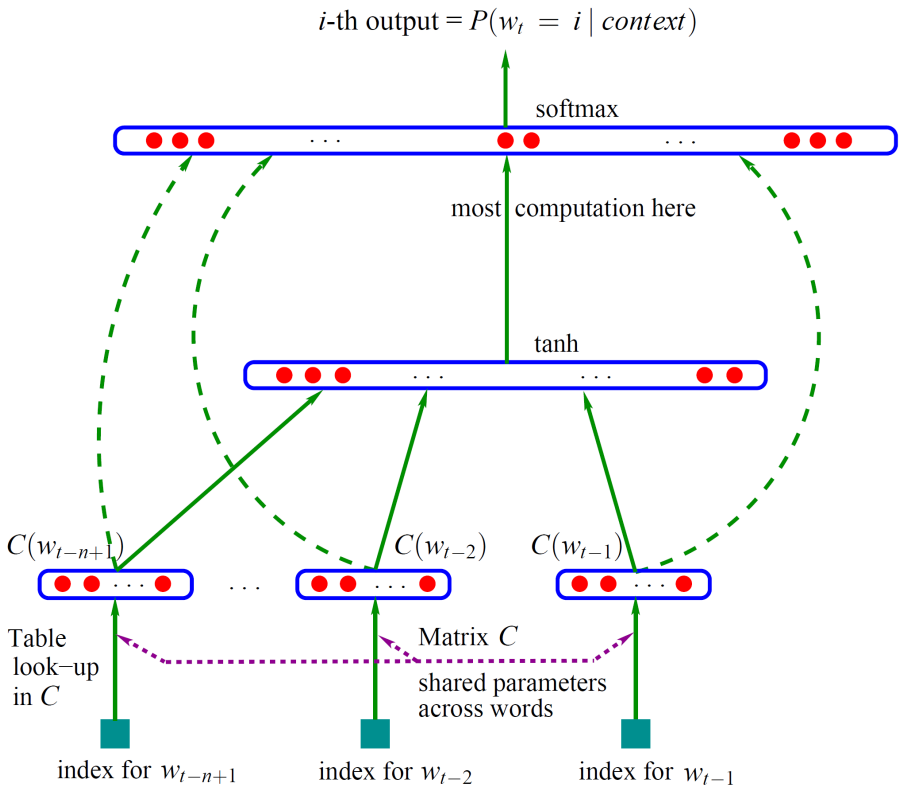

Figure 2.9: Classic neural language model (Bengio et al., 2003).

Throughout the years, different methods for training word embeddings have been developed. One of the most popular methods, known as the Word2Vec model, was introdued by Mikolov et al. (2013). The Word2Vec model uses one of the following two approaches for training the embeddings: Continuous Bag of words (CBOW), and Skip-gram. In the CBOW model, the embedding is created by using the words surrounding a target word as input and the target word itself as the output. In the Skip-gram model, however, an opposite approach is used to create and extract an embedding vector with the target word being fed as input, and the surrounding words as output. Both CBOW and Skip-gram architectures have an input, projection, and output layers. After the training is complete, the weights that connect the input layer to the projection layer will be considered as the embedding matrix.

GloVe is another method for training word embeddings, developed by Pennington et al. (2014). As opposed to the Word2Vec model which directly utilizes instances of word co-occurrences, GloVe uses statistics of word co-occurrances, making semantic relationships between words more explicit.

A more recent word embedding model, ELMo, was developed by Peters et al. (2018). Trained using a task of bidirectional language modeling (i.e. language modeling in both forward and backward fashions), ELMo is different from the previous embeddings in two major ways. Firstly, ELMo is contextual, meaning that the embedding for a given word might vary based on the different contexts in which the word occurs. Secondly, ELMo is character-based, which means that the model does not have a pre-defined vocabulary of words used for training, but it rather extracts the word embeddings from the constituent characters of the words. These differences can make the word embedder more robust and generalizable to downstream tasks.

In our work, the use of GloVe and ELMo embeddings has been experimented with.

2.3 Evaluation Metrics

An essential part of developing a classification system is evaluating how it performs when exposed to new samples. As a result, appropriate evaluation metrics that provide a reliable measure of the system's performance are needed. In this section, a brief summary of the most common evaluation metrics used for classification is presented.

Most evaluation metrics are based on a confusion matrix that gives information about the system's prediction for the samples and also the true label for the samples in question. Figure 2.10 shows a typical confusion matrix for a binary classification problem.

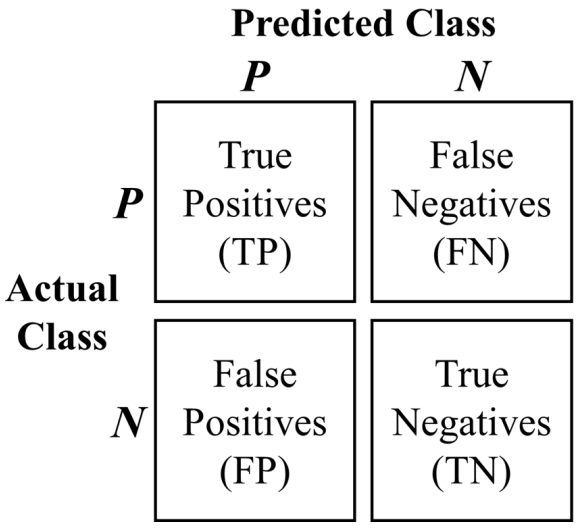

Figure 2.10: The visualization of a confusion matrix for binary classification.

2.3.1 Accuracy

The accuracy measure takes into account the number of correctly classified samples out of all predictions made by the system, irrespective of the samples' class. It is calculated using Equation 13.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

In the case of a classification system aimed at detecting health issues, accuracy is not a reliable measure, since there is often the necessity to give more weight to positive labels (i.e. at-risk users). As a result, more informative measures, such as precision, recall, and F-measure are preferred.

2.3.2 Precision

Precision measures the ratio of correctly classified positive samples (true positives) out of all the samples classified as positive and is calculated using Equation 14.

$$precision = \frac{TP}{TP + FP} \tag{14}$$

2.3.3 Recall

The recall measure is used to calculate the ratio of correctly detected positive samples out of all the positive samples, both detected and undetected by the system, in the data set. Equation 15 is used to measure recall.

$$recall = \frac{TP}{TP + FN} \tag{15}$$

In health applications, recall is a very important metric that measures the capacity of a system to emit alerts when needed.

2.3.4 F-Measure

There is a trade-off between precision and recall, causing one to decrease as the other increases. A system that has a tendency to predict more negative labels can have a good precision but a low recall, while a system that outputs too many positive labels has a good recall, but a low precision.

The F-measure was developed in an attempt to combine both precision and recall into a single measure, making it possible to do model optimization and selection based on a single evaluation metric.

$$F_\beta = (1 + \beta^2) \times \frac{precision \times recall}{(\beta^2 \times precision) + recall} \tag{16}$$

In Equation 16, the β parameter is used to control the importance of precision over recall and give more weight to one instead of the other. In this thesis, F1 score is used for evaluation purposes (i.e. β=1), placing equal emphasis on precision and recall. This leads to Equation 17.

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{17}$$

2.3.5 Pearson Correlation

The Pearson correlation coefficient (Spearman, 1987) can act as a measure for both regression and classification problems and is calculated using Equation 18.

$$r = \frac{\sum_{i=1}^{n}(T_i - \bar{T})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(T_i - \bar{T})^2 \sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{18}$$

Assuming a binary classification task, in Equation 18, T_i and Y_i stand for the true and predicted values (which can be either 0 or 1) for the i-th sample. $\bar{T}$ and $\bar{Y}$ represent the average of all true and all predicted labels, respectively. Finally, n is the number of samples for which this measure is calculated.

2.3.6 Macro-average versus Weighted-average versus Micro-average

When classification is performed on data with more than two classes, evaluation metrics can be calculated using two different methods, either of which can be chosen based on the use case. The first method is macro-averaging, which calculates a certain metric (e.g. precision, recall, or F-measure) for each individual class versus the other classes separately, and then reports the average over all classes. Therefore, all classes contribute equally to a macro-average score, regardless of the number of samples in each class. Equation 19 shows how the macro-average score for a specific metric is calculated, where S_i represents to the score on the specific class i, and N_c represents the number of classes.

$$Macro\text{-}average\ S = \frac{1}{N_c}\sum_{i=1}^{N_c} S_i \tag{19}$$

The second method, weighted-averaging, takes into account the number of samples in each class when calculating the average score for all classes. Weighted-averaging leads to different contributions from different classes based on the class distribution in the data. Equation 20 is used to compute a weighted-average score, where S_i represents the score on the specific class i, N_c represents the number of classes, and $n(c_i)$ represents the number of samples in class i.

$$Weighted\text{-}average\ S = \frac{\sum_{i=1}^{N_c} n(c_i)S_i}{\sum_{i=1}^{N_c} n(c_i)} \tag{20}$$

Finally, micro-averaging is done by first summing up the number of true positives, false positives, true negatives, and false negatives over the set of different classes, and using the original formula to calculate a score, hence calculating one single global score for all classes. If calculated on all classes, the micro-average F1 score is mathematically equal to micro-average precision, micro-average recall, and accuracy.

In this thesis, macro and micro-averaging will be used.

2.4 Emotion Classification and Mental Health Assessment

2.4.1 Emotion Detection

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.

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 had 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) (see Section 2.2.1) 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 included 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 an LSTM (Hochreiter and Schmidhuber, 1997) (see Section 2.2.1). Using the dataset of SemEval 2018 task 1 (Affect in Tweets) (Mohammad et al., 2018), they achieved their best Pearson 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[7]. 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 CNN (LeCun et al., 1999) (see Section 2.2.1). 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.

2.4.2 Mental Health Assessment

Many researchers have used corpora from Twitter, Facebook, Reddit, blogs and online forums as resources to experiment with classification tasks pertaining to mental health issues (Calvo et al., 2017).

Pestian et al. (2010) experimented with different machine learning methods for suicide note classification. The features used in the study included words, part of speech tags, readability scores, and emotions. The best accuracy of 74% was achieved by a logistic regression model.

DeVault et al. (2013) studied the symptoms of psychological distress in dialogues with a virtual agent. The use of a Naïve Bayes classifier for the detection of post-traumatic stress disorder (PTSD) and distress yielded a 20% improvement over the baseline accuracy of 53.5%, and showed that the automatic assessment of psychological distress is indeed possible.

More recently, Jackson et al. (2017) used clinical texts obtained through the Clinical Record Interactive Search[8] to extract symptoms of severe mental illness. The authors made use of TextHunter (Ball et al., 2014) (a natural language processing information extraction tool) and an SVM classifier, and were able to classify 38 symptoms with an F1-score of 85%.

Shen and Rudzicz (2017) used different feature sets including Word2Vec embedding, latent

[7] https://csn.cancer.org/
[8] https://crisnetwork.co

Dirichlet allocation topic modelling, lexico-syntactic features, and N-grams (unigrams and bigrams) to detect anxiety in Reddit posts. Initially, the authors compared the results achieved by an SVM and a 2-layer neural network. Though both classifiers performed well, the SVM yielded marginally better results. However, they achieved their best result of 98% accuracy using the neural network with n-gram probabilities and word embeddings combined with Linguistic Inquiry and Word Count (LIWC) features (Pennebaker et al., 2007).

Coppersmith et al. (2014) explored the automatic detection of post-traumatic stress disorder (PTSD), depression, bipolar disorder, and seasonal affective disorder (SAD) in Twitter data, using LIWC features and character and word N-grams, and found the latter resulting in superior performance.

Benton et al. (2017) used multi-task learning to predict suicide risk and a variety of mental health conditions from Twitter data, including anxiety, depression, PTSD, and schizophrenia. It was found that a multi-task framework can be effectively used in cases with limited data.

Apart from individual efforts, shared tasks (e.g. (Coppersmith et al., 2015; Milne et al., 2016; Lynn et al., 2018; Zirikly et al., 2019)) have also been organized to encourage the development of common benchmarks (datasets and metrics) and the comparison of approaches for the detection of distress in online textual data.

All of the previous work described above used a classic classification approach that does not measure how early the detection is performed. The eRisk shared task (Losada et al., 2017) was created with the goal of addressing issues related to early risk detection of mental health problems. Early detection can be used in many applications where intervention is needed and the timing of the intervention is important. Examples of such applications are identifying potential sexual offenders or people with suicidal inclinations (Losada et al., 2017). According to Losada et al. (2017), risk assessment approaches are currently focused at detection after the fact, i.e. ringing an alarm after the harmful behavior has occurred, while the timing of the detection can be of utter importance. Therefore, it is important to try to minimize the time between seeing the first sign of a harmful behavior and triggering an alarm. Having that in mind, the eRisk organizers have created this shared task in order to encourage the development of risk detection approaches that can model the risk process. The eRisk shared task has also introduced evaluation metrics specifically designed to

take into account the delay in risk detection.

The first eRisk shared task (Losada et al., 2017) was held with the aim of the early detection of depression, using a dataset of chronologically recorded online posts from depressed and non-depressed users. For the first time, Early Risk Detection Error (ERDE) was used as an evaluation metric that takes into account the delay in detection by awarding early decisions and penalizing late ones (Losada and Crestani, 2016). Since it was the first time that such an evaluation metric was used, most of the teams that took part in the shared task focused on making accurate, albeit delayed detection, with the highest F1-score being 64% and the lowest $ERDE_{50}$ score[9] being 9.68% (Losada et al., 2017).

The second eRisk shared task (Losada et al., 2018) included two tasks: Early risk detection of depression and early risk detection of anorexia. Like the year before, the ERDE evaluation metric was used as the main metric alongside F1, precision, and recall (Losada et al., 2018). Trotzek et al. (2018) designed the best performing models for both tasks. In the depression task, their system which utilized a logistic regression classifier with bag of words as input features, achieved an F1-score of 64% and an $ERDE_{50}$ of 6.44%. Using a convolutional neural network (CNN) (LeCun et al., 1999) model with fasttext word embeddings (Joulin et al., 2017; Bojanowski et al., 2017; Mikolov et al., 2018) as input features, they achieved an F1-score of 85% and an $ERDE_{50}$ of 5.96% in the anorexia task. Their interesting approach and promising results inspired the ensemble model presented in this thesis and motivated us to take part in the CLPsych and eRisk shared tasks, the results of which will be presented in Chapters 4 and 5.

[9] A detailed description of $ERDE_o$, where o is either 5 or 50, can be found in Chapter 5.

Chapter 3

Contextual Emotion Detection

Our first text classification model was developed within the framework of the EmoContext 2019 international shared task (Chatterjee et al., 2019b), which is focused on the detection of emotions in context.

In this chapter, we present the task of contextual emotion detection, the dataset used for the task, and the architecture of our model. Based on the official results, the effects of different model components and their effectiveness in the classification task are analyzed.

3.1 EmoContext

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.

It is worth mentioning that, in many cases, context plays an irreplaceable role in the identification of the emotion present in a piece of text. As an example, consider the sentence *I have tears in my eyes*. If taken out of context, the emotion in this sentence will most likely be classified as *sad*.

However, if the same sentence is part of the following conversation:

Speaker 1: We did it! We finally won the championship!

Speaker 2: Finally!

Speaker 1: I have tears in my eyes.

taking context into account, the last sentence uttered by *speaker 1* has *happy* as its main emotion class.

Although much research has focused on the detection of emotions in tweets and blog posts (e.g. Mohammad, 2012; Desmet and Hoste, 2013; Liew and Turtle, 2016), emotion detection in dialogues, as well as in single utterances has received very little attention. This topic can have significant impact for the development of social chatbots, with the aim of creating an emotional connection between a user and a chatbot (Banchs, 2017).

Task 3 of SemEval 2019 (*EmoContext*) has focused on contextual emotion detection over 4 classes: *happy*, *sad*, *angry*, and *others* (Chatterjee et al., 2019b). We participated in this shared task under the name *CLaC Lab* and used a combination of artificial neural networks (see Section 2.2) with recurrent units and attention mechanism, and SVM (see Section 2.1) to address this multi-class classification task.

3.2 Dataset and Task

The dataset 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 3.1 shows two samples of the dataset[1].

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 3.1: Two sample dialogues from the EmoContext 2019 dataset.

The goal is to detect the emotion of speaker 1 in turn 3, taking into account the previous turns.

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

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 3.2 summarizes some statistics of the 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 3.2: Statistics of the EmoContext 2019 dataset.

3.3 The Model

Figure 3.1 shows the 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.

3.3.1 The Neural Feature Extractor

As shown in Figure 3.1, the neural feature extractor is a recurrent neural network (see Section 2.2.1) 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 (see Section 2.2.2) 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 21 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 22, where y_t is the

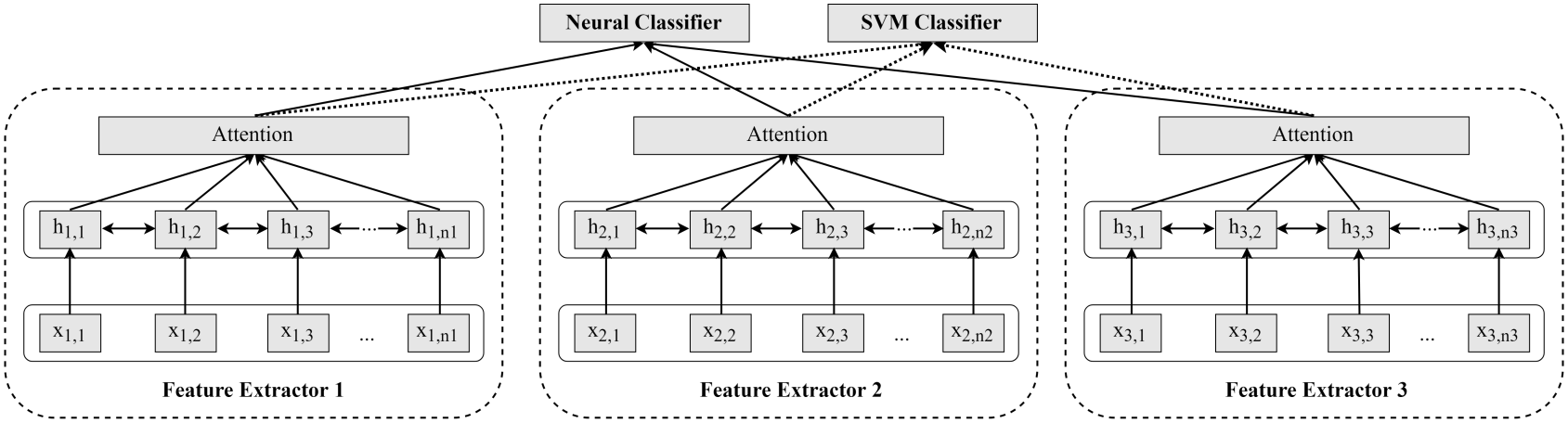

Figure 3.1: Architecture of the model for EmoContext 2019.

output of the hidden layer at time-step t, and f_y is the function that calculates the output value based on h_t.

$$h_t = f_h(x_t, h_{t\pm1}) \tag{21}$$

$$y_t = f_y(h_t) \tag{22}$$

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 23 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).

$$Attention = \sum_{t'=1}^{n} y_{t'}\omega_{t'} \tag{23}$$

In our proposed model, the weights are calculated by applying a single N-to-1 feed-forward layer on 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 24 and 25 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.

$$\nu_t = y_t \times w \tag{24}$$

$$\omega = Softmax([\nu_1, \nu_2, \nu_3, \ldots, \nu_n]) \tag{25}$$

3.3.2 The Classifier

As shown in Figure 3.1, we experimented with two types of classifiers at the output layer: A fully-connected neural network (see Section 2.2.1), followed by a softmax activation function, and an SVM (see Section 2.1), 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.

3.4 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 with for the task of contextual emotion detection.

3.4.1 Word Embeddings

In order to test our model, we experimented with two different pretrained word embeddings (see Section 2.2.2). 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[2] (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

[2]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.

lead to better results in our task (see Section 3.5).

3.4.2 POS Tags

The spaCy library[3] 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.

3.4.3 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.

3.4.4 Neural Network Optimization

The Adam optimizer (Kingma and Ba, 2015) with a learning rate of 10^{-4} was used to train the neural 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 3.2), 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.

3.4.5 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

[3] https://spacy.io/

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.

3.4.6 Overall Training Process

As indicated in Section 3.3.2, 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.

3.5 Results

The official evaluation metric used at the EmoContext shared task is the micro-average F1 score (see Section 2.3.4) over the three main emotion classes, i.e. *happy*, *angry*, and *sad* (ignoring the 4th class, *others*).

Figures 3.2 and 3.3 show the performance of each model on the development dataset, throughout the training process. As Figures 3.2 and 3.3 show, 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.

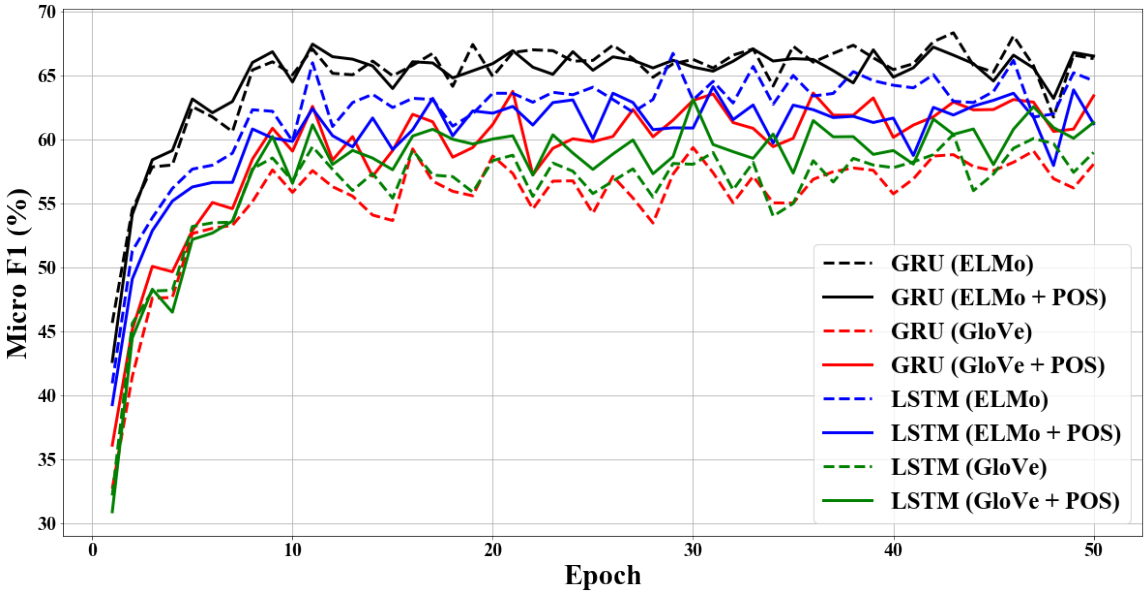

Figure 3.2: Performance of the model at EmoContext 2019 (in micro-average F1 score over the three main emotion classes) on the development dataset, as a function of the number of training epochs with the **neural classifier**.

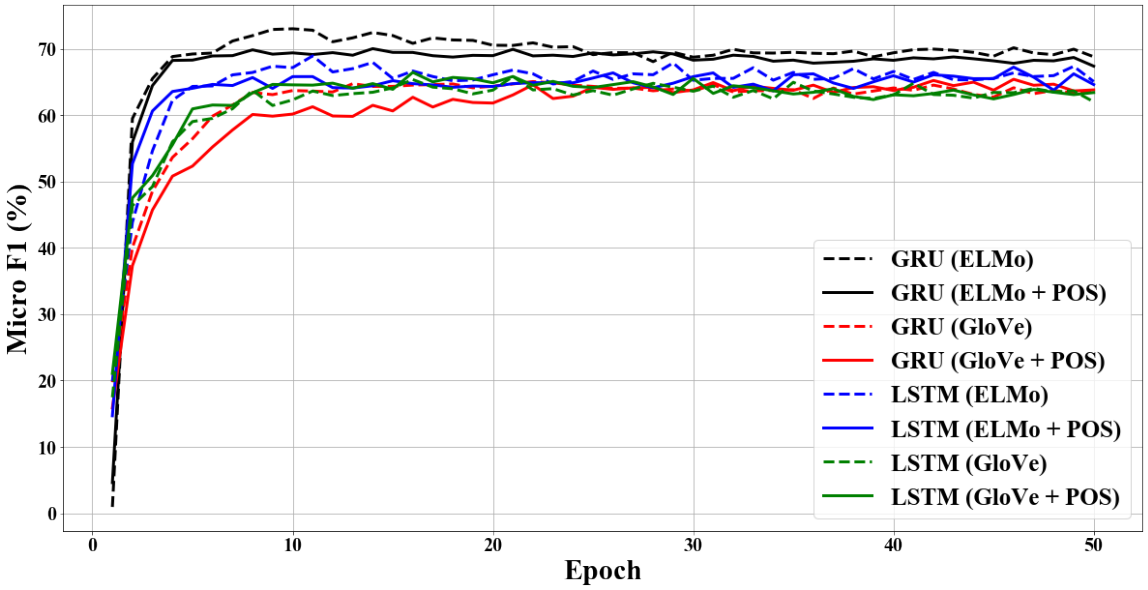

Figure 3.3: Performance of the model at EmoContext 2019 (in micro-average F1 score over the three main emotion classes) on the development dataset, as a function of the number of training epochs with the **SVM classifier**.

The notation *<type-of-recurrent-unit>+<type-of-classifier> with <type-of-input-features>* is used in the rest of this chapter, 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 3.4.6, 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[4].

Table 3.3 shows the performance of each model[5], 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*.

The results also show that, with the exception of the two models *LSTM+NN with GloVe* and *GRU+NN with GloVe* which have inferior performance compared to the baseline system on the test dataset, all the other models significantly outperform the baseline model on both development and test data.

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

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

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 3.3: The performance of each model on the EmoContext 2019 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.

3.6 Discussion

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

3.6.1 Effect of Input Features

The results in Table 3.3 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 3.3 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 3.3 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.

3.6.2 Effect of the Recurrent Units

The results in Table 3.3 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.

3.6.3 Effect of the Classifiers

Table 3.3 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, Figures 3.2 and 3.3 show 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 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.

3.6.4 A Closer Look at the Emotion Classes

The row labelled *AVERAGE* in Table 3.3 provide information regarding the difficulty of detecting each class. Table 3.3 shows that, among the three main classes, *happy*, *angry*, and *sad*, the class *happy* was the most difficult to detect.

Table 3.2 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 3.4.4) 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.

3.6.5 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 3.4 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.

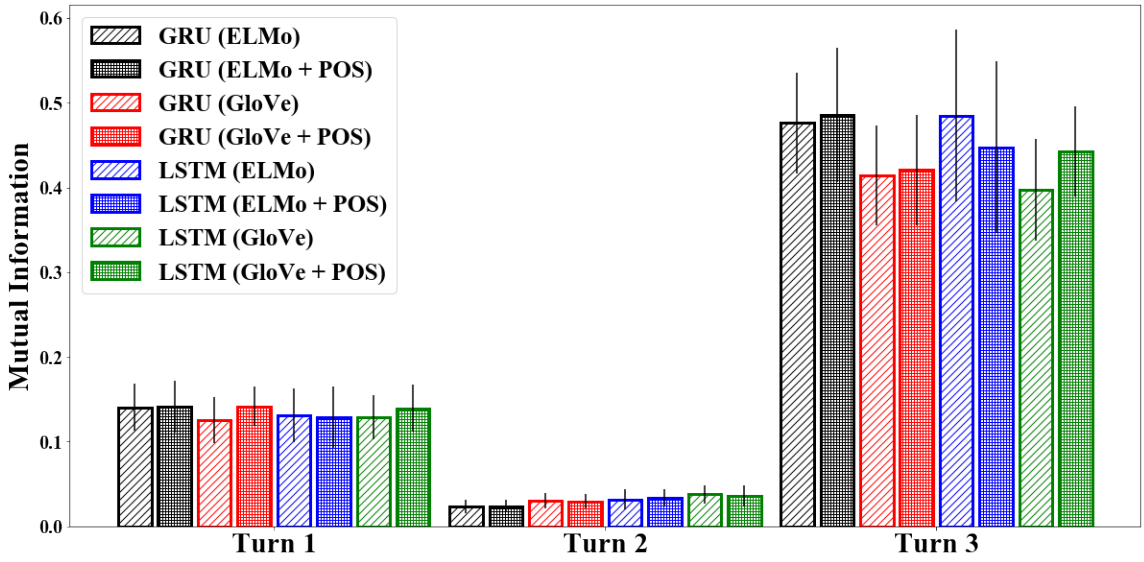

Figure 3.4: The average mutual information between the features of each dialogue turn, extracted by each neural feature extractor, and the classes.

As Figure 3.4 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 3.4 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.

3.7 Further Analysis

Since 311 teams submitted their results to EmoContext 2019, all of which cannot be presented in this chapter, we only describe overall statistics of the results and present some popular approaches utilized by the top performing teams.

As shown in Table 3.4, our best model performed approximately 3% above average. However, some of the submitted models by other teams that have achieved good results can be analyzed to improve our classification system.

	Micro-average F1
Max	79.59%
3rd Quartile	73.17%
CLaC	70.72%[6]
Median	69.40%
Mean	65.99%
1st Quartile	63.70%
Min	1.43%

Table 3.4: Performance statistics of all participants (taken from Chatterjee et al. (2019b)).

Looking at the models developed by the top performing teams in the EmoContext shared task, it can be concluded that ensemble models have been found useful for text classification.

Similar to our approach, Basile et al. (2019) experiment with an LSTM-based model that feeds the 3 turns of conversation into 3 different branches. The model uses attention at the end of each branch to take into account the most interesting parts of each conversation turn before combining them. As input features, they pretrain their own embeddings on the English Wikipedia. In an attempt to deal with unbalanced data, they develop a second model which does a two-stage classification: first classifying a sample as *emotional* or *other*, after which if the sample is recognized as *emotional*, there is a second classification that labels the sample as *happy*, *angry*, or *sad*. They also experiment with two other models using transfer learning: The 3rd model uses a universal sentence encoder (Cer et al., 2018) and the 4th model utilizes Bert (Devlin et al., 2019). In order to combine the strengths of different approaches, the output softmax probabilities of all 4 models are concatenated and fed to a final classifier. Their best F1 score of 0.77 on the test set is achieved by an ensemble model with an SVM classifier at the end.

Huang et al. (2019) create an ensemble model which is made up of two deep learning sub-models. The first model is a hierarchical recurrent model which has two recurrent components,

[6]The result that we submitted to the EmoContext shared task was slightly higher than the results reported in Table 3.3, which was achieved by also using the development dataset to train the SVM classifier.

called *encoder* RNN and *context* RNN. Using a combination of pretrained GloVe and ELMo embeddings, the *encoder* RNN is responsible for mapping conversations turns to vectors and the *context* RNN processes those vectors, thus being able to represent the multi-turn structure of the conversations. The hierarchical recurrent model alongside a pretrained Bert model constitute the ensemble model which achieves an optimal performance with an F1 score of 0.77.

Liang et al. (2019) experiment with pretrained word embeddings by Baziotis et al. (2017) as well as Google's pretrained Word2Vec embeddings as input features to a CNN with max pooling that processes conversation turns in a parallel but separate fashion. This architecture is used to train classifiers for 3 different sub-tasks. The first classifier is trained to distinguish between the four different classes. The second classifier is aimed at classifying only the three emotional classes *happy*, *angry*, and *sad*, while the last classifier is trained to do a binary *others-or-not* classification. The final class is decided by using voting to comibine the labels output by the three classifiers. Their experiments show that the ensemble model performs better than the individual classifiers, achieving an F1 score of 0.76.

An interesting finding by Basile et al. (2019) is that the contextual emotion detection task can be challenging even for humans. To measure the difficulty of the task, two fluent English speakers were asked to manually annotate 300 samples taken from the development set and measured their F1-scores against the gold labels. Their F1 scores were calculated to be 0.73 and 0.72, showing challenges in classifying *others* versus the emotional classes and also in distinguishing between *sad* and *angry* classes. Therefore, it can be concluded that it is possible to take advantage of the strengths and the information provided by different models by using them in an ensemble architecture, as well as leveraging the extra information provided by transfer learning through using different types of pretrained embeddings.

In this chapter, we proposed a model for the task of contextual emotion detection. We evaluated our model with the EmoContext 2019 shared task dataset 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%.

The main takeaway from the experiments reported in this chapter is that neural networks can act as great feature extractors (something that SVMs are not capable of handling on their own) while SVMs proved to be better final classifiers than neural networks. We used this finding in our subsequent experiments that are reported in Chapters 4 and 5.

Chapter 4

Suicide Risk Assessment

In the previous chapter, we developed a model for the task of contextual emotion detection in short conversations. The model consisted of a gated recurrent neural network with attention that took as input ELMo embeddings alongside one-hot POS tags and acted as a deep neural feature extractor and an SVM as the final classifier. Combining the power of deep neural networks in extracting relevant features and the power of SVMs in classification proved useful in improving the results of a non-hybrid architecture. We also observed from the results of other teams that training multiple models and combining their outputs can be helpful in final classification, since different models can contribute to classification by extracting different features.

In this chapter, we extended our model from Chapter 3 and developed an ensemble text classification model in the framework of the CLPsych 2019 shared task (Zirikly et al., 2019) and test it using the test data and the evaluation metrics provided by the organizers. The main reason for using an ensemble approach is the limited number of annotated samples available for this task in comparison to the number of training samples available for the model developed in Chapter 3, a limitation that can lead to overfitting when a single neural model is utilized.

In the following sections, first an overview of the task and the dataset is given. In Section 4.2, the model's architecture, the implementation, and the different configurations used for the experiments are presented in detail. Section 4.3 includes the results achieved in the CLPsych 2019 shared task and presents an analysis.

4.1 Dataset and Task

The CLPsych 2019 shared task (Zirikly et al., 2019) focuses on the prediction of a person's degree of suicide risk based on a collection of their Reddit posts (Shing et al., 2018). It is a multi-class classification task where a subject can be assigned to one of the four categories of *no* (class *a*), *low* (class *b*), *moderate* (class *c*), or *severe* risk (class *d*), and consists of three different tasks:

Task A aims at suicide risk prediction based solely on the posts written on the Suicide Watch subreddit[1].

Task B focuses on making the same prediction by taking into account a person's posts on Suicide Watch, as well as their posts on other subreddits.

Task C has the goal of estimating suicide risk by looking at a subject's posts on different subreddits, but excluding Suicide Watch.

type of dataset	**sub-task**	**# of users**				**average # of posts per user**
		class a	**class b**	**class c**	**class d**	
Train	Task A	127 (26%)	50 (10%)	113 (23%)	206 (41%)	1.85
	Task B	127 (26%)	50 (10%)	113 (23%)	206 (41%)	115
	Task C	624 (63%)	50 (5%)	113 (11%)	206 (21%)	56.4
Test	Task A	125				1.49
	Task B	125				76.9
	Task C	248				57.4

Table 4.1: Distribution of labels in the CLPsych 2019 dataset for different sub-tasks.

[1] https://www.reddit.com/r/SuicideWatch

The first two tasks are dedicated to assessing risk; while Task C aims at screening. We participated in all 3 tasks[2] under the team name *CLaC* and ranked first in tasks A and C.

Table 4.1 shows some statistics of the dataset. As shown in the table, in tasks A and B, most of the samples belong to class *d*, while class *b* has a significantly smaller number of samples. In task C, the bulk of the data belongs to class *a*, leading to a bias towards this class. As a result, the official evaluation metric defined for this task excludes class *a* (see Section 4.3).

4.2 System Overview

Our system is composed of 8 neural network sub-models, each with a specific type of input word embedding and hidden layer. The extracted neural features and softmax probabilities from all 8 neural networks are combined by a fusion component and the resulting features are used in the final SVM classifier. Figure 4.1 illustrates the overall architecture of the system. Each component is explained in the following sections.

4.2.1 Sub-models

As shown in Figure. 4.1, each sub-model includes an input layer that receives as input the posts by a user, and vectorizes its tokens using an embedding layer. The output of the input layer is then fed to a hidden layer, which is followed by a post-level attention/pooling layer that creates a representation of the post from its constituent tokens. The user-level attention layer is responsible to calculate the vector representation of the user, using her/his online posts. Finally, the output (classification) layer predicts the probability distribution of the four classes.

Our main focus during the development of the sub-models was to include diversity of information sources, so that the final ensemble model can incorporate different points of views when performing the final classification.

[2]This research was recognized as an IRB exempt by Concordia University's research ethics board.

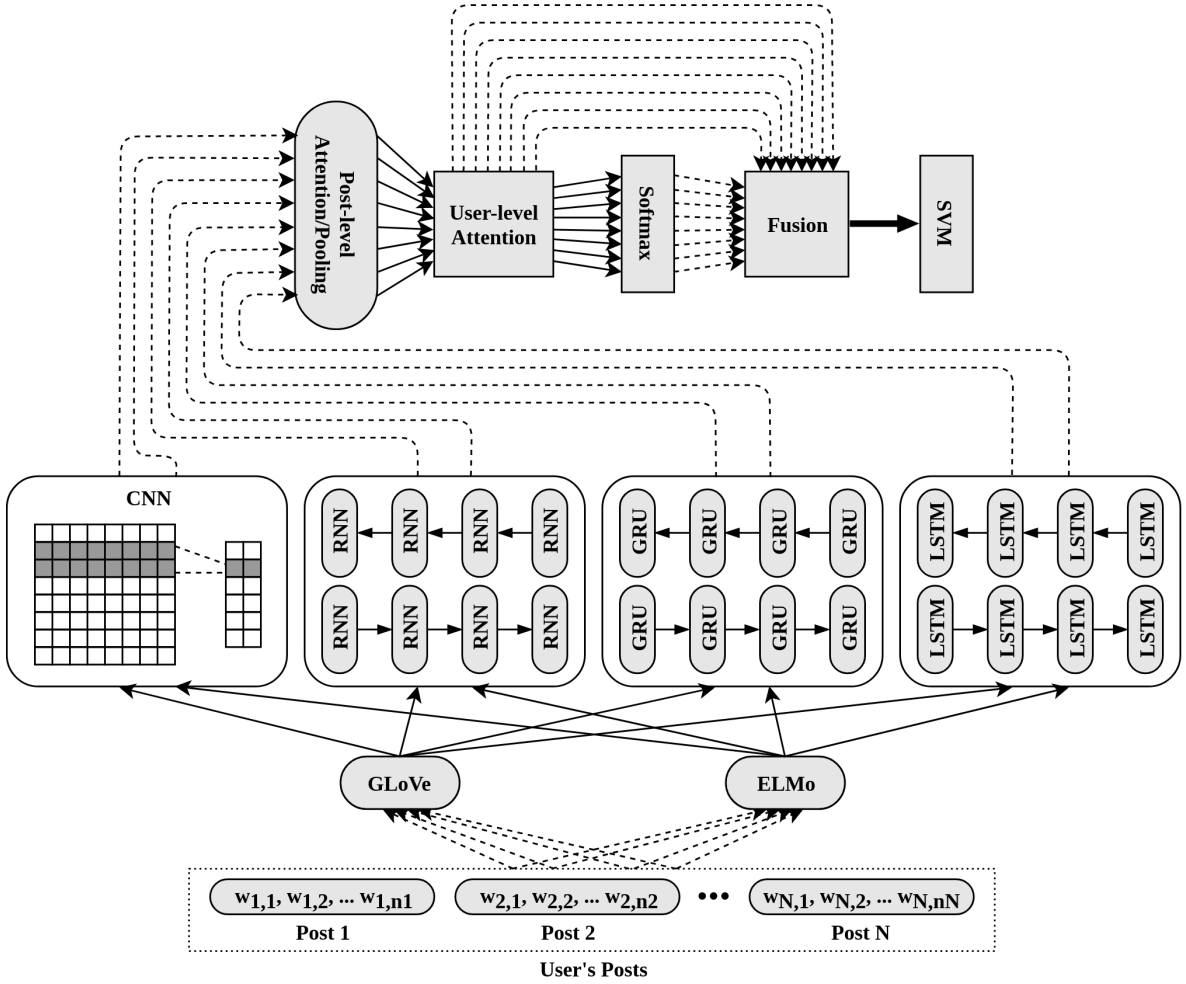

Figure 4.1: Architecture of the model for CLPsych 2019. The number of arrows between components corresponds to the number of sub-models that move in that flow. The rounded-corner boxes represent the components that work at the post level, while the sharp-corner boxes are user-level components. The solid lines represent neural connections; while the dotted lines show the flow of data without the existence of a neural connection. The bold arrow between the Fusion and SVM corresponds to the flow of data that exists only in the final model.

Input Layer

The inputs to the model are the online posts of each user. Each post is first tokenized, and the tokens are sent to the word embedder, in order to be converted into dense vectors. As shown in Figure 4.1, these token vectors are then fed to the hidden layer.

As we did in Chapter 3, two different pretrained word embeddings were experimented with. The first word embedder was the 300d version of GloVe (Pennington et al., 2014) that was pretrained on 840B tokens of web data from Common Crawl. The second word embedder was the original 1024d version of ELMo (Peters et al., 2018), which was pretrained on the 1 Billion Word Language Model Benchmark (Chelba et al., 2014). These two word embeddings were used in order to provide our ensemble model with sub-models that utilize both contextual (ELMo) and non-contextual (GloVe) word embedders in their input layer.

Hidden Layer

The hidden layer is responsible for processing the token vectors, generated by the input layer. As shown in Figure 4.1, we have experimented with four hidden architectures in our sub-models: a CNN (LeCun et al., 1999) that processes token n-grams separately, and a Bidirectional Vanilla RNN (BiRNN), a Bidirectional Long Short-Term Memory (BiLSTM) (Hochreiter and Schmidhuber, 1997) and a Bidirectional Gated Recurrent Unit (BiGRU) (Cho et al., 2014), all of which process token vectors sequentially, from first to last and vice-versa, by taking into account the preceding and following tokens, respectively.

Post-level Attention/Pooling Layer

In order to create a vector representation for each post, three different types of pooling were applied to the output of the hidden layer. In the rest of this chapter, these will be referred to as *AVG*, *MAX*, and *ATTN*.

AVG pooling simply averages the output vectors of the hidden layers. *MAX* pooling is applied on the resulting vectors after applying Concatenated Rectified Linear Unit (CReLU) on the output vectors of the hidden layers (i.e. ReLU applied on the concatenation of each output vector and its

negative). *ATTN* is an attention mechanism (Bahdanau et al., 2015) applied to the output vectors of the hidden layers. While *ATTN* may not be considered a pooling method, we do so in order to differentiate between *ATTN* and the attention mechanism presented in Section 4.2.1. Since *ATTN*'s functioning is similar to the attention mechanism used to calculate the weighted average of the representations for a user's posts, its mechanism will be explained in detail in Section 4.2.1.

User-level Attention Mechanism

It was hypothesized that all posts by a user do not contribute equally to signal her/his mental state. In order to take into account the posts of each user based on their importance in detecting suicide risk, an attention mechanism was used. This mechanism automatically assigns weights to each post from a user, then calculates the weighted average of the representations of all the posts, and uses this average as a representation of the user. Equation 26 shows how the output of the attention mechanism is computed.

$$U = \sum_{i'=1}^{N} p_{i'} \omega_{i'} \tag{26}$$

where $p_{i'}$ stands for the representation of the i'-th post by a user, $\omega_{i'}$ refers to the weight assigned to the post, and U corresponds to the vector representation for that specific user.

In order to calculate the corresponding weights for the posts, a single n-to-1 fully connected layer is first applied to the representation of each post, where n corresponds to the size of the document representation. The final weights are calculated by applying a softmax to the concatenation of the results of applying the fully-connected layer on the representations of all posts from a user. Equations 27 and 28 show how the weights are calculated:

$$\nu_i = p_i \times w \tag{27}$$

$$\omega = Softmax([\nu_1, \nu_2, \nu_3, \ldots, \nu_N]) \tag{28}$$

where w corresponds to the weights in the neural layer, and ν_i refers to the resulting scalar, after feeding p_i (the representation of the i-th post) to the fully-connected layer.

As stated in Section 4.2.1, the overall mechanism of *ATTN* is similar to the attention mechanism

applied to a user's posts. The only difference resides in the level of their functioning: the attention mechanism is applied to the post representations, whereas *ATTN* is applied to the outputs of the hidden layer, at (multiple-)token-level.

Output (Classification) Layer

The final layer in the sub-models is a feed-forward fully-connected layer that maps the output of the user-level attention to a vector with size 4 (corresponding to the four classes). At the end of this layer, a softmax activation function gives as the output, the predicted probability distribution over the classes *negative* and *positive*.

The Sub-models' Optimization Technique

Similarly to Chapter 3, PyTorch (Paszke et al., 2017) was used to develop and train the neural sub-models. Each sub-model was trained separately on the training data and optimized using the validation data.

The Adam optimizer (Kingma and Ba, 2015) with a learning rate of 5×10^{-4} was used as the optimization technique. Cross-entropy was used as the loss function, and in order to handle the imbalanced class distribution, weights were assigned to each class proportional to the inverse of the number of samples in that class. Due to limitation in computational resources, mini-batches with a maximum size of 32 were applied at the post level for each user.

4.2.2 Ensemble Model

As shown in Figure 4.1, the ensemble model is composed of several neural sub-models, a fusion component, and a final SVM classifier.

The Fusion Component

The fusion component is responsible for creating a final vector representation for each user from the neural features and the predicted probability distributions over classes.

The neural features of the user representations are the result of each sub-model's attention component. In the fusion components, these user representations are first concatenated, and later, the

mutual information between each neural feature and the final classes is calculated (using the Scikit-learn library (Pedregosa et al., 2011)). A subset of these features that have the highest mutual information with the final classes are then selected as the final neural features.

The fusion component also uses the predicted probability distributions of the classes for each user from the softmax output of all sub-models. The final user representations are generated by concatenating the neural features and the predicted probability distributions from all sub-models, to be fed to the SVM (see Figure 4.1).

The Support Vector Classifier

As shown in Figure 4.1, the final classifier is an SVM, which uses as input the final user representations generated by the fusion component. The SVM was trained on the samples from the training data, and the validation dataset was used to find the best set of hyperparameters. We used the Scikit-learn library (Pedregosa et al., 2011) for developing and training the SVM model. The final hyperparameters of the SVM classifiers are presented in Section 4.2.3.

4.2.3 Final Submitted Models

Before training the model and its sub-models, we randomly sampled posts from 33% of the users in the training dataset in a stratified fashion, in order to be used for validation.

When feeding the posts to the sub-models, only the first 200 or 400 tokens were used[3], depending on which limit yielded a better performance at validation time, and the rest were disregarded.

The training process of each sub-model was stopped when the performance on the validation data was at its maximum (for each task, we used the main evaluation metric for that specific task; see Section 4.3). The validation data was also used in order to find the best set of hyperparameters of the models for each task.

The full model utilizes 8 different sub-models, each one with a unique input word embedding (GloVe or ELMo) and hidden layer type (CNN, Bi-RNN, Bi-LSTM or Bi-GRU). Table 4.2 shows the hyperparameters of the sub-models for each task, where each sub-model is named by its type of hidden layer and input word embedding.

[3]The average size of posts across all tasks is ~78 tokens.

For each task, we submitted three different runs:

Run 1 where the SVM classifier only uses the neural features.

Run 2 where the SVM classifier only uses the predicted probability of classes.

Run 3 where both the neural features and predicted probabilities are used by the SVM classifier.

Table 4.3 summarizes the hyperparameters used in each run.

	Task A				Task B				Task C			
	#HL / KH	#HN / #K	Pooling Type	Max Post Length	#HL / #K	#HN / KH	Pooling Type	Max Post Length	#HL #K	#HN / KH	Pooling Type	Max Post Length
CNN–GloVe	2	300	MAX	400	2	200	AVG	400	2	100	MAX	400
CNN–ELMo	1	400	MAX	400	2	100	MAX	400	2	100	MAX	400
Bi-RNN–GloVe	2	64	MAX	400	2	32	ATTN	200	2	32	ATTN	200
Bi-RNN–ELMo	2	32	MAX	400	1	64	ATTN	200	1	64	ATTN	400
Bi-LSTM–GloVe	2	32	AVG	400	1	64	ATTN	200	2	32	ATTN	200
Bi-LSTM–ELMo	2	32	AVG	400	1	64	ATTN	200	1	64	ATTN	200
Bi-GRU–GloVe	2	64	MAX	400	2	32	ATTN	200	2	32	ATTN	400
Bi-GRU–ELMo	2	64	MAX	400	1	64	ATTN	200	1	64	ATTN	400

Table 4.2: Hyperparameters used for each sub-model at CLPsych 2019. *#HL*: number of hidden layers, *#HN*: number of hidden nodes in each layer, *KH*: kernel height (for the CNNs), *#K*: number of kernels (for the CNNs).

	Task A					Task B					Task C				
# of Neural Features	174					80					925				
SVM's Hyperparameter	kernel	degree	γ	C	class weight	kernel	degree	γ	C	class weight	kernel	degree	γ	C	class weight
Run 1	poly	1	auto	3.0	yes	sigmoid	–	scale	0.8	no	poly	3	scale	0.1	yes
Run 2	poly	4	scale	0.1	no	poly	2	scale	0.5	yes	sigmoid	–	scale	0.4	yes
Run 3	poly	1	auto	0.3	yes	sigmoid	–	scale	0.2	no	poly	2	scale	0.2	yes

Table 4.3: Hyperparameters used in the submitted runs at CLPsych 2019. The column *degree* refers to the degree of the polynomial kernels. The values of *auto* and *scale* for γ refer to when the parameter γ is set to 1/number-of-features and 1/(number-of-features×variance-of-features), respectively. The value of *class weight* indicates whether weights proportional to the inverse of the number of samples from classes are applied to the parameter *C*.

4.3 Results and Discussion

Table 4.4 presents a summary of the results of the three runs in each of the three tasks, based on three evaluation metrics (see Section 2.3 for more details on these metrics):

Macro-F1: Macro-averaged F1 on classes *a*, *b*, *c*, *d* for tasks A and B, and macro-averaged F1 on classes *b*, *c*, *d* for task C. This was the official metric for this shared task, on which we optimized our systems.

Flagged: Precision, recall and F1 for *flagged* versus *non-flagged*, where *flagged* includes classes *b*, *c*, *d*, and *non-flagged* consists of class *a*.

Urgent: Precision, recall and F1 for *urgent* versus *non-urgent*, where *urgent* includes classes *c* and *d*, and *non-urgent* consists of classes *a* and *b*.

In tasks A and C, the highest macro-averaged F1 was achieved by run 3, and for Task B, the highest F1 was achieved by run 2. This shows the effectiveness of using both the neural features and the predicted probabilities for the final SVM classifier.

In all three tasks, the best flagged F1 was achieved by run 1, showing that using only the neural features leads to better performance when distinguishing between no-risk users (class a) and users that require attention (classes b, c, d).

Figures 4.2, 4.3, and 4.4 show the confusion matrices for the three primary systems submitted for tasks A, B, and C, respectively. As evident in the confusion matrices for tasks A and B, there has been some difficulty in distinguishing between classes *c* and *d*, which is to be expected as the two classes both include samples with higher risk. In task C, however, the most number of mistakes happen by misclassifying samples belonging to classes *d* and *c* as class *a*. Although this sounds surprising as classes *a* and *d* are very different and at two different sides of the continuum, we suspect that this can be due to the distribution of labels for task C: with 63% of the training data belonging to class *a*, the trained model can be more biased towards labeling samples as belonging class *a*. This implies that, alongside using weighted loss to deal with the unbalanced dataset, other measures that take into account the distance between classes can be used to improve the classification.

		Accuracy	Macro-F1	flagged			urgent		
				P	R	F1	P	R	F1
Task A	**run 1**	0.504	0.481	**0.957**	**0.890**	**0.922**	0.738	0.819	0.776
	run 2	0.512	0.416	**0.957**	0.881	0.918	**0.963**	0.762	**0.851**
	run 3	**0.560**	**0.533**	**0.957**	**0.890**	**0.922**	0.838	**0.838**	0.838
Task B	run 1	0.408	0.359	**0.871**	0.844	**0.857**	0.688	**0.743**	0.714
	run 2	**0.456**	**0.381**	0.806	0.824	0.815	**0.750**	0.714	**0.732**
	run 3	0.416	0.339	0.839	**0.848**	0.843	0.700	0.737	0.718
Task C	run 1	0.669	0.250	**0.602**	**0.767**	**0.675**	0.537	**0.705**	0.610
	run 2	0.669	0.239	**0.602**	0.747	0.667	**0.562**	0.682	0.616
	run 3	**0.673**	**0.268**	**0.602**	0.757	0.671	**0.562**	0.703	**0.625**

Table 4.4: Results of each run on for tasks A, B, and C on the CLPsych 2019 test dataset. The primary runs (the ones considered in the ranking) are shown in between asterisks and the best results are highlighted in bold.

		class a			class b			class c			class c		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1
Task A	*run 1*	**0.656**	**0.840**	**0.737**	0.385	0.179	0.244	0.429	**0.375**	0.400	0.481	0.625	0.543
	run 2	0.625	0.833	0.714	0.000	0.000	0.000	**0.536**	0.278	0.366	0.558	0.617	0.586
	run 3	**0.656**	**0.840**	**0.737**	**0.462**	**0.300**	**0.364**	0.464	0.371	**0.413**	**0.577**	**0.667**	**0.619**
Task B	run 1	0.531	**0.586**	**0.557**	**0.231**	0.136	0.171	0.179	0.185	0.182	0.500	0.553	0.525
	run 2	0.500	0.471	0.485	0.154	**0.286**	**0.200**	**0.250**	**0.280**	**0.264**	**0.615**	0.542	**0.577**
	run 3	**0.562**	0.545	0.554	0.077	0.062	0.069	0.179	0.192	0.185	0.538	**0.560**	0.549
Task C	run 1	**0.890**	**0.789**	**0.836**	**0.077**	0.083	0.080	0.143	0.167	0.154	0.442	**0.622**	0.517
	run 2	0.877	0.786	0.829	**0.077**	**0.111**	**0.091**	0.071	0.111	0.087	**0.519**	0.562	**0.540**
	run 3	0.884	0.787	0.833	**0.077**	0.100	0.087	**0.179**	**0.200**	**0.189**	0.462	0.615	0.527

Table 4.5: Breakdown of results from Table 4.4 for each class separately. The primary runs are shown in between asterisks and the best results are highlighted in bold.

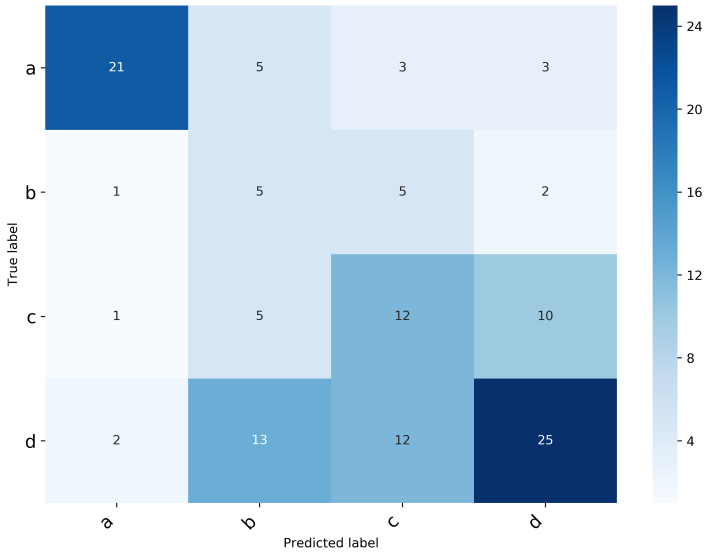

Figure 4.2: Confusion matrix of results for the primary system of task A of CLPsych 2019.

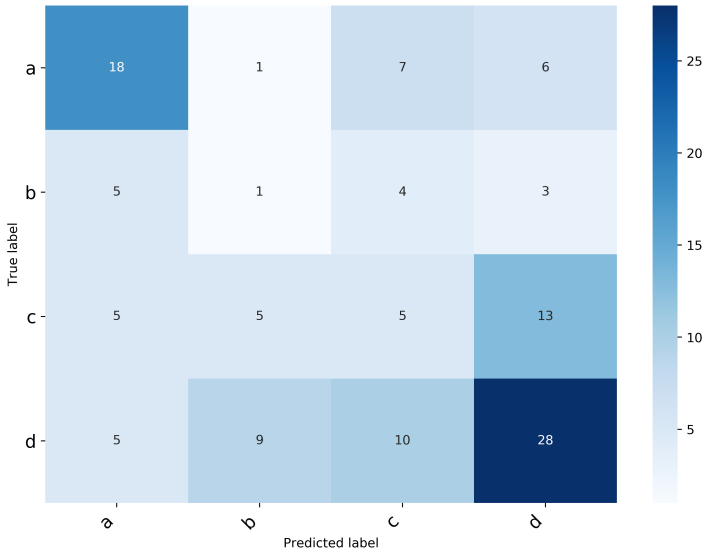

Figure 4.3: Confusion matrix of results for the primary system of task B of CLPsych 2019.

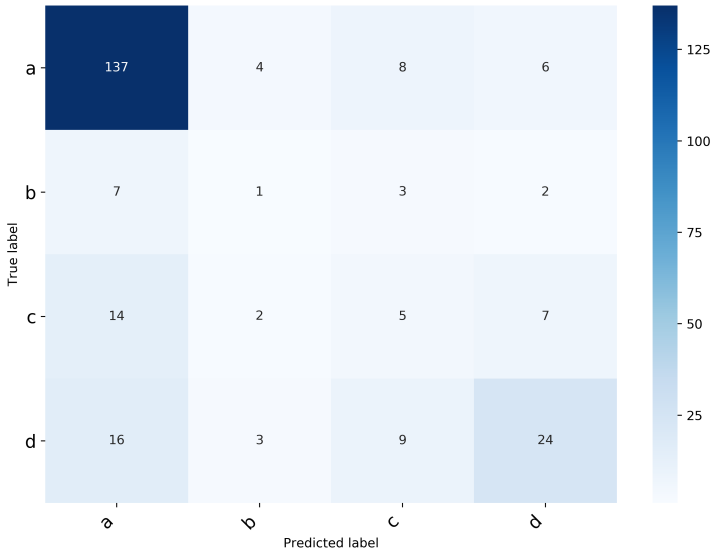

Figure 4.4: Confusion matrix of results for the primary system of task C of CLPsych 2019.

Further Analysis

As mentioned in Section 4.1, tasks A and B are focused on risk assessment while task C is about screening. The difference between tasks A and B is that in task B, for a given user, in addition to the user's Suicide Watch posts, there is extra data available which is taken from other subreddits in which the user has been active. Although additional information is expected to improve the final model, results from different teams' participation to the two tasks show that, in most cases, using posts from other subreddits leads to a worse performance (Zirikly et al., 2019).

An interesting finding by Matero et al. (2019) is that using a dual-context approach can be helpful when data is available from both Suicide Watch and other subreddits for a given user. Using Bert embeddings as input, their developed model consists of two RNNs that process posts taken from Suicide Watch and Non-Suicide Watch subreddits separately before aggregating them to output the final label, leading to the best reported result among participating teams in task B. However, it is important to consider that in a real-life scenario in any social platform, it often happens that more relevant posts are mixed with less relevant ones. Therefore, the need for a system that can automatically assign more weight to important posts without using metadata seems inevitable.

Another system that was able to obtain interesting results in task B was developed by Ruiz et al. (2019). In an effort to develop a knowledge-rich approach, they utilize a variety of tools such as the Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES) (Savova et al., 2010), the Word-Emotion Association Lexicon (Mohammad and Turney, 2013), and the Stanford CoreNLP sentiment annotator (Manning et al., 2014) to extract handcrafted features. Topic modeling and semantic role labeling are also used and a Doc2Vec model is built to extract 300-dimensional vectors that are representative of the Reddit posts. The extracted features are then used as input to an ensemble classifier of NB, SVM, and Gradient Boosting (GB). The results of the ensemble model show the effectiveness of using handcrafted features to identify more important posts for the final classification.

In this chapter, we extended the model developed in Chapter 3 and proposed a model based on an ensemble technique that uses a fusion of neural features and predicted probability distributions over classes from 8 neural sub-models, with an SVM as a final classifier. Our first rank in tasks A and C of CLPsych 2019 shared task shows that this technique can be useful in the task of suicide risk assessment. Moreover, it was found that using both neural features and predicted probability of classes generally led to a better performance. In the next chapter, we use the same ensemble approach for a similar task, but from a different angle: the early detection of anorexia risk from social media posts. We evaluate the effectiveness of our model in performing a task which is focused at making accurate decisions as well as early ones.

Chapter 5

Early Risk Detection of Anorexia

In the last chapter, we presented our approach to the task of suicide risk assessment. The model consisted of 8 neural sub-models which were used to extract neural features and predict class probabilities. The extracted features and predicted probabilities were then used by an SVM classifier which would label a sample as belonging to one of the 4 risk categories, ranging from no risk to high risk. The results of our participation to the CLPsych 2019 shared task motivated us to do further experiments and evaluations of the developed model.

In this chapter, we used the same ensemble approach presented in Chapter 4, for the early detection of signs of anorexia and evaluate our model in the framework of the eRisk 2019 task 1 (Losada et al., 2019), using the evaluation metrics specifically developed for this shared task to take into account the timing of the decisions as well as their correctness. For this reason, in this chapter, we inspect the capabilities of our approach in making decisions having seen as little evidence as possible, contrary to having seen all of a user's posts before making a decision for that user.

The rest of the chapter is organized as follows: Section 5.1 provides an overview of the task and the data set used. In Section 5.2, the evaluation metrics used for early risk detection are presented. Sections 5.3 and 5.4 are dedicated to a brief overall architecture and a more detailed description of model variants that were employed for the experiments, respectively. Finally, Section 5.5 includes a summary and discussion of the results.

5.1 Task and Dataset

Following the success of the eRisk 2018 task 2 (Losada et al., 2018), the eRisk 2019 task 1 (Losada et al., 2019) focuses on the early detection of anorexia in online posts. The data used for the task is a collection of Reddit users labelled as anorexic or non-anorexic (Losada and Crestani, 2016), along with a collection of their Reddit posts, recorded chronologically.

For the training phase, the data from the previous year (eRisk 2018 task 2), including both training and test sets, was made available. For the testing phase, posts were released on an item-by-item basis in chronological order for a new collection of Reddit users. The goal was to detect users suffering from anorexia, having observed as few posts from them as possible, keeping in mind that emitting a positive label is irreversible. That is to say that once a user is classified as having anorexia, the system cannot change the predicted label to negative after observing more evidence.

Considering the fact that the timing of producing an alert is also of significance, in addition to precision, recall, and F1-score, two other metrics were used: Early Detection Error (ERDE) measure which penalizes late decisions, and latency-weighted F1, a modified version of F1 score that takes into account the delay of the decision[1].

Table 5.1 shows some statistics of the datasets. As shown in the table, the datasets are highly imbalanced, with about 90% of the users not suffering from anorexia.

Dataset	**Source**	**Positive**	**Negative**	**All**
Training	Train 2018	20 (13%)	132 (87%)	152
Validation	Test 2018	41 (13%)	279 (87%)	320
Testing	–	73 (9%)	742 (91%)	815

Table 5.1: Distribution of user labels in the datasets. The 2018 datasets refer to the eRisk 2018 task 2 data.

[1]The details of the evaluation metrics for eRisk 2019 task 1 is explained in Losada et al. (2019).

5.2 Evaluation Metrics

As standard evaluation metrics of precision, recall, and F-score (see Section 2.3) do not take into account the delay of a system before making decisions, new measures have been introduced by the eRisk shared task organizers. These measures have been developed specifically for this shared task and are designed to take into account the time between making an observation and emitting a decision. In this section, these new evaluation metrics are presented.

5.2.1 ERDE

ERDE (Losada and Crestani, 2016) is the error measure designed for the task of early risk detection. It gives a penalty for the delay in making a correct positive decision (correctly classifying a user as at risk) and as expected, the penalty increases with the delay. This error measure is meant to minimize the time between observing posts from an at-risk user and producing an alert. The following equation is used to calculate ERDE (Losada et al., 2017) for a specific binary decision *d* (i.e. the final classification of a single sample on the test data):

$$ERDE_o(d) = \begin{cases} c_{fp} & \text{if } d\text{=positive AND ground truth=negative (FP)} \\ c_{fn} & \text{if } d\text{=negative AND ground truth=positive (FN)} \\ lc_o(k).c_{tp} & \text{if } d\text{=positive AND ground truth=positive (TP)} \\ 0 & \text{if } d\text{=negative AND ground truth=negative (TN)} \end{cases}$$

As shown in the equation, there are different penalties for the 4 different cases in a binary classification:

- The parameter c_{fp} is the penalty for a false positive is equal to the proportion of positive labels in the test set (which in case of eRisk 2019 task 2 is approximately 0.089).

- The parameter c_{fn} is the penalty for a false negative is equal to 1, which is much greater than the previous case, i.e. a false positive, because most of the samples belong to the negative class. Defining a big penalty for a false negative can help prevent the system from producing too many negative labels as a result of being trained on too many negative samples.

- $lc_o(k).c_{tp}$ is used to measure the penalty for a true positive based on the number of posts that were observed before deciding on the class of a user. $lc_o(k)$ is obtained using Equation 29 and c_{tp} is set to 1.

$$lc_o(k) = 1 - \frac{1}{1 - e^{k-o}} \tag{29}$$

In Equation 29, k is the number of evidences observed before emitting a correct positive label and o is the parameter that, when plotted, decides where on the X axis, the cost sharply increases (see Figure 5.1). The parameter o is set to either 5 or 50, resulting in 2 different versions of ERDE, referred to as ERDE_5 and ERDE_{50}.

- There is no error measure for making a correct but late classification of negative samples. The reason behind this is that the focus of this error measure is on the early identification of users at risk.

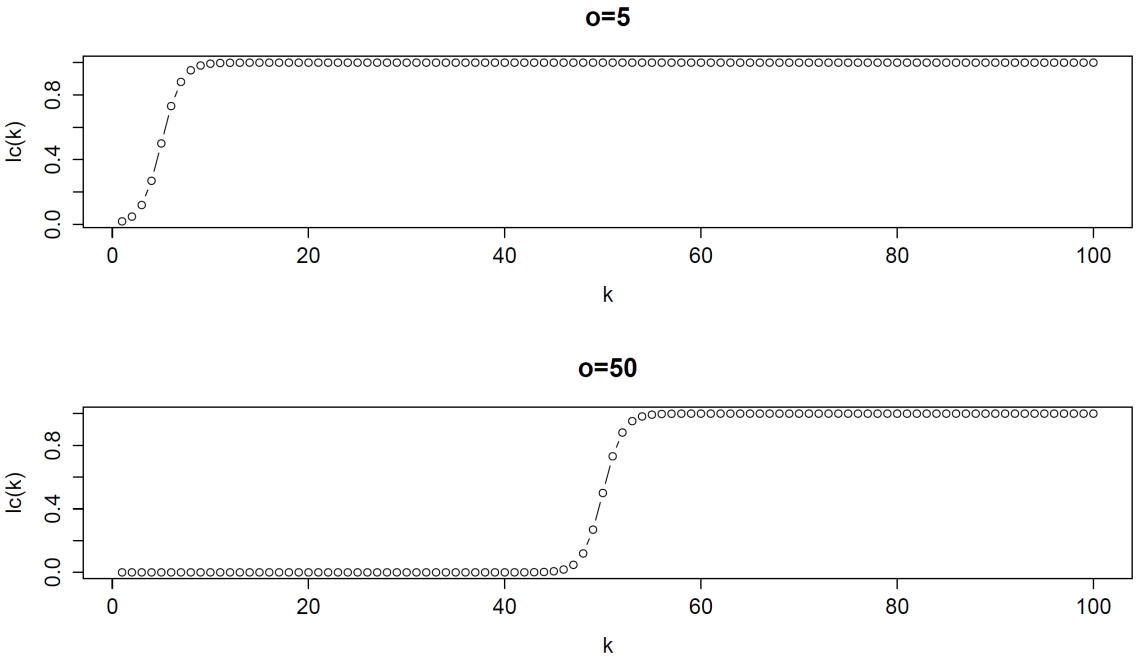

Figure 5.1: Latency cost functions $lc_5(k)$ and $lc_{50}(k)$ (Losada et al., 2017)

Although the ERDE measure has advantages over traditional metrics, some disadvantages of this measure were identified by Losada et al. (2019), among which the following can be mentioned:

- The cost of making a delayed but correct positive classification increases very quickly (see Figure 5.1), making it difficult to distinguish between different systems.
- A correct positive label that is produced instantly (after having the first chunk of evidence) still receives some penalty.
- Although it is possible to compare different systems based on ERDE, it is difficult to properly evaluate and make sense of the performance of a single system by only having its ERDE at hand.

5.2.2 Latency-Weighted F1 Score

In order to make up for the disadvantages of ERDE, a new measure called *latency-weighted F1 score* was proposed (Losada et al., 2019). By multiplying the original F1 score by a new metric of *speed*, the latency-weighted F1 score is capable of capturing the correctness as well as the delay of a decision (see Equation 30).

$$latency\text{-}weighted\ F1 = F1.speed \tag{30}$$

As shown in Equation 31, the *speed* parameter is calculated by taking into account the median of the penalty terms assigned to all true positive decisions:

$$speed = (1 - \text{median}\{penalty(k_u) : u \in U, d_u = g_u = 1\}) \tag{31}$$

In the equation above, d_u refers to the system's decision for a user and g_u refers to the user's golden truth. $penalty(k_u)$ stands for the penalty term calculated for a true positive emission by the system, having observed *k* evidences from the user.

The penalty for the delay of each true positive decision is calculated using equation 32, where the *p* parameter determines the speed with which the penalty goes up.

$$penalty(k_u) = -1 + \frac{2}{1 + e^{-p.(k_u - 1)}} \tag{32}$$

The latency-weighted F1 score has a number of advantages over the ERDE measure, including

more inseparability and a milder penalty increase. Furthermore, a perfect system which identifies all true positives, having observed only the first round of evidence, receives a perfect latency-weighted F1 score of 1.

5.3 The Model

The model that is developed and used in this chapter is similar to the one explained in Chapter 4 except for the following differences:

Firstly, for all the CNN-based sub-models, max pooling was used over the CReLU activation function, while for the sub-models utilizing recurrent components (including BiRNN, BiGRU, BiLSTM), attention was used.

Secondly, as opposed to the model in Chapter 4 where a subset of the neural features with the highest mutual information with classes were selected, in the current model, all of the neural features are utilized.

These two changes were made as a result of experiments on the held-out validation data.

5.4 Experimental Setup

This section describes our experiments with the above model for our participation to the eRisk 2019 shared task (Losada et al., 2019).

5.4.1 Sub-models Implementation

Again, *PyTorch* (Paszke et al., 2017) was used to implement and train the sub-models. The Adam optimizer (Kingma and Ba, 2015) was used, and the learning rate was set to 5×10^{-4}. Cross-entropy was used as the loss function, in order to handle the imbalanced distribution of the positive and negative classes in the training set (see Table 5.1), weights proportional to the inverse of the number of samples of each class were assigned to that class. Due to lack of computational resources, mini-batches with a maximum size of 128 were used at the post level for each user and only the first

100 tokens of the posts were used[2]. In order to minimize the amount of padding in the batches, posts with similar number of tokens were assigned to the same batch.

In order to fine-tune the other hyperparameters of the sub-models (including the number and size of convolutional filters, number of recurrent units, and number of training epochs), each sub-model was individually trained with the training set and optimized on the validation set (see Table 5.1), based on the F1 score. The specifics of the 8 different sub-models are shown in Table 5.2. Since each sub-model is composed of a unique pair of hidden layer and word embedding type, they will later be referred to as <*hidden-type*>-<*embedding-type*> (see the second column of Table 5.2).

#	Name	Hyperparameters
1	*CNN-GloVe*	100 bigram convolution filters, trained for 10 epochs
2	*CNN-ELMo*	200 unigram filters and 50 bigram convolution filters, trained for 6 epochs
3	*BiRNN-GloVe*	one layer of 64 vanilla RNN units, trained for 14 epochs
4	*BiRNN-ELMo*	one layer of 50 vanilla RNN units, trained for 13 epochs
5	*BiLSTM-GloVe*	one layer of 32 bidirectional LSTM units, trained for 31 epochs
6	*BiLSTM-ELMo*	one layer of 64 bidirectional LSTM units, trained for 14 epochs
7	*BiGRU-GloVe*	one layer of 64 bidirectional GRUs, trained for 14 epochs
8	*BiGRU-ELMo*	one layer of 64 bidirectional GRUs, trained for 8 epochs

Table 5.2: Hyperparameter values used in the 8 sub-models at eRisk 2019.

5.4.2 Ensemble Classifiers

Scikit-learn (Pedregosa et al., 2011) was used to develop the SVM classifier used in the ensemble model. Three different versions of ensemble classifiers were developed:

(1) ***Ens-Feat*** is the version of the ensemble model that only utilizes the neural features. The SVM classifier in this version uses a sigmoid kernel. The γ and C parameters in the SVM were set to *auto* (i.e. 1/<number-of-features>) and 4, respectively.

(2) ***Ens-Prob*** uses only the predicted class probabilities from the softmax activation function at

[2]This limit only truncated a small number of posts, as the average length was ~37.47 tokens in the eRisk 2018 task 2 data.

the end of the neural sub-models. It utilizes a polynomial kernel with the degree of 1. The γ and C parameters in the SVM were set to *scale* (i.e. 1/[<number-of-features>×<variance-of-features>]) and 1, respectively.

(3) ***Ens-All*** utilizes both neural features and predicted class probabilities in its SVM classifier, that uses a sigmoid kernel, and has its values of γ and C set to *auto* and 2, respectively.

5.4.3 Submitted Runs

Based on the results with the validation set, 5 runs were submitted to the shared task server. For the 1st and 2nd runs, *CNN-GloVe* and *CNN-ELMo* were used, respectively, as stand-alone models[3], and *Ens-Feat*, *Ens-Prob*, and *Ens-All* comprised the 3rd, 4th and 5th runs.

5.5 Results and Discussion

Table 5.3 shows the official results of our submissions, as well as selected runs from other teams (as reported in Losada et al. (2019)) that achieved the best result with one of the official evaluation metrics, or achieved competitive results. For the results of our team (*CLaC*), we indicate in Table 5.3 the specific name of the models used in the five submitted runs.

As shown in Table 5.3, the model *Ens-All* achieved the highest F1 (0.7073) and latency-weighted F1 (0.6908) scores of all participants' runs. This was in line with our intuition that using an ensemble model that makes use of both neural features and predicted class probabilities from the 8 sub-models has a higher capability of detecting the correct class after observing a small number of writings. The results also show that the *CNN-ELMo* model can achieve F1 and latency-weighted F1 scores that are competitive to *Ens-All*, and outperforms *Ens-Feat* and *Ens-Prob* in these two metrics. The *CNN-ELMo* model also resulted in the best recall, $ERDE_5$ and $ERDE_{50}$, showing the potential of this model to be used independently for the task of early risk detection of anorexia.

Table 5.3, also shows that all our models, except *CNN-GloVe* (run 0) achieved significantly superior performances in terms of F1 score and latency-weighted F1 (teams *lirmm* and *INAOE-CIMAT* achieved the next best F1 and latency-weighted F1 scores). Run 1 of team *lirmm* achieved

[3]These two sub-models achieved the most promising results among all the sub-models, during the training phase.

team	model	run	#writings	P	R	F1	$ERDE_5$	$ERDE_{50}$	l-w F1
CLaC	CNN-GloVe	0	109	0.4463	0.7400	0.5567	0.0672	0.0393	0.5437
CLaC	CNN-ELMo	1	109	0.6061	0.8219	0.6977	0.0573	0.0312	0.6895
CLaC	Ens-Feat	2	109	0.6020	0.8082	0.6900	0.0602	0.0313	0.6766
CLaC	Ens-Prob	3	109	0.6292	0.7671	0.6914	0.0627	0.0355	0.6752
CLaC	Ens-All	4	109	0.6374	0.7945	**0.7073**	0.0625	0.0343	**0.6908**
lirmm		0	2024	0.74	0.63	0.68	0.09	0.05	0.63
lirmm		1	2024	**0.77**	0.60	0.68	0.09	0.06	0.62
Fazl		2	2001	0.09	**1.00**	0.16	0.17	0.11	0.14
UNSL		0	2000	0.42	0.78	0.55	**0.06**	0.04	0.55
UNSL		4	2000	0.31	0.92	0.47	0.06	**0.03**	0.46
INAOE-CIMAT		3	2000	0.67	0.68	0.68	0.09	0.05	0.63

Table 5.3: Official results on the first task of the eRisk 2019 shared task. *#writings*: maximum number of writings (Reddit posts) that were processed for a user, *P*: Precision, *R*: Recall, *l-w F1*: Latency-Weighted F1 score.

the highest precision. The best recall was achieved by run 2 of the team *Fazl*. Runs 0 and 4 of the team *UNSL* achieved the highest $ERDE_5$ and $ERDE_{50}$, respectively, where we could achieve competitive results using *CNN-ELMo*.

We believe that one of the most important components in all of our developed sub-models is the user-level attention mechanism that enables the sub-models to automatically assign weights to user posts proportional to their relevance, thus contributing to the good results achieved in this task.

The number of writings processed by the models submitted by each team shows that our models used a significantly lower number of writings in comparison to the other teams[4]. This shows that our systems have a great potential of making early and correct decisions. This is supported by an even larger gap between the latency-weighted F1 scores of our team and the runs submitted by other teams, in comparison to the gap in F1 score.

Although our systems achieved the best or competitive results according to different evaluation metrics, we suffered from lack of computational resources when running the models that use the ELMo embedder for around 2000 iterations. The models had to be run for approximately 2000 times due to the item-by-item release of the test data which was chosen for the eRisk 2019 shared task (in the previous eRisk shared tasks, the test data was released in 10 chunks, making the number of iterations equal to 10). Despite this technical drawback, the advantages of using ELMo to extract context-sensitive embeddings greatly outweigh its disadvantages. This can also be observed by comparing the results achieved by *CNN-GloVe* and *CNN-ELMo*.

In this chapter, we further experimented with the ensemble model which was developed in Chapter 4. Once again, the effectiveness of this model was evaluated in the context of the eRisk 2019 task 1, whose focus was the early detection of anorexia risk. The results of our submitted system as well as a comparison between different systems' results show that our approach can indeed be useful in making correct as well as early decisions to identify users that are at the risk of anorexia.

[4]The average number of writings processed by the participating teams was 1273.

Chapter 6

Conclusion and Future Work

6.1 Summary

This thesis explored the development of an automatic approach which can be used in order to detect emotions and also different types of distress in textual data.

In Chapter 3, we experimented with different deep architectures and embeddings to develop a model for the task of emotion detection in small 3-turn conversations. Our experiments showed that using contextual word embeddings such as ELMo as input features instead of non-contextual ones such as GloVe can significantly improve the model's performance. In the case of using GloVe embeddings, we also saw benefits in using POS tags as they help model contextual information which is absent in the embeddings. It was also observed that, in almost all of our experiments, the models using SVM as the classifier outperformed the models using neural classifiers by a large margin. The takeaway from the work presented in this chapter was the shown strength of character-based embeddings and the potential improvement as a result of using deep neural networks for feature extraction and an SVM for final classification.

Consequently, we used our findings in emotion detection to develop a model aimed at the assessment of suicide risk in online posts. As mentioned above, in Chapter 3, it was determined that ELMo embeddings can have an advantage over GloVe embeddings and that replacing a neural classifier with an SVM on features extracted by a neural network can improve the classification. As a result, in Chapter 4, we made use of both of these findings in the development of a classification

model to assess suicide risk in online data. Since we encountered the usual challenge of having access to limited labelled data, we hypothesized that using several types of feature extractors can help capture more information from the data available. This lead to the development of an ensemble model which utilizes 8 different neural sub-models, each composed of a unique pair of word embeddings and deep architecture. These neural sub-model were used to extract neural features and their own predictions of class probabilities leading to a more rich and diverse set of features which could in turn be leveraged for a better classification. We used an SVM for the final classification part, having observed the potential of an SVM classifier when it is fed the features extracted by neural models. It was also determined that, in general, feeding the SVM both neural features and predicted class probabilities leads to improved results. Using this ensemble architecture, we achieved the first rank in tasks A and C of the CLPsych 2019 shared task which motivated us to test this model in another similar setting with a different focus, the results of which have been documented in Chapter 5.

Chapter 5 described our experiments and results in applying the developed ensemble model from Chapter 4 to the task of early risk detection of anorexia. The focus of the task in question was not only to make a correct detection of at-risk users but also being able to do so having observed as few posts from a given user as possible. This provided us with the opportunity to evaluate the model from another interesting and useful perspective. Although the model was not specifically trained to do early predictions, it performed surprisingly well, ranking first in the first task of eRisk 2019 in terms of both F1 score and latency-weighted F1, which also takes into account the delay in making predictions.

The results in Chapters 4 and 5 led us to conclude that such an ensemble architecture which makes use of diverse sources of information is indeed helpful in the detection of distress. Our results also show that such a model can be used to make up for the lack of a large annotated dataset, on which end-to-end deep learning models heavily rely.

6.2 Contributions

This thesis presented a number of contributions:

- The implementation of different deep learning models for the task of contextual emotion detection and participation to the EmoContext shared task in SemEval 2019 (Mohammadi et al., 2019a), and a performance analysis of the emotion detection models and the different components used (Mohammadi et al., 2019d) (see Chapter 3).

- The implementation of an ensemble model including several attention-based deep models for the task of suicide risk assessment and achieving state-of-the-art results in tasks A and C of the CLPsych 2019 shared task (Mohammadi et al., 2019b) (see Chapter 4).

- The evaluation of the ensemble model for the task of early risk detetcion of anorexia and achieving state-of-the-art results in the first task of eRisk 2019 (Mohammadi et al., 2019c) (see Chapter 5).

6.3 Future Work

Possible extensions and improvements to our work include the following:

Contextual Emotion Detection In Chapter 3, we discussed the development and evaluation of a deep learning model for the task of contextual emotion detection in short dialogues. Three future directions can be proposed:

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

Secondly, the use of different number of neural features for each dialogue turn can be studied. As shown in Figure 3.4, 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 amount of 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.

Detection of Distress in Online Data In Chapters 4 and 5, we presented an ensemble approach which can be used to detect distress in the social media posts of a user. The ensemble model achieved top performances at both CLPsych 2019 (Mohammadi et al., 2019b) and eRisk 2019 (Mohammadi et al., 2019c) shared tasks. We believe that the user-level attention mechanism has played an important role in the high results achieved on these shared tasks. It would be interesting to qualitatively analyze the results of the attention mechanism, to see how they correlate with human perception, i.e. whether the posts to which the attention mechanism assigns more weights are actually the same posts that seem more informative to a health specialist for detecting anorexia.

Also, during the development phase, it was determined that removing each of the 8 sub-models (even the sub-models with low individual performances) negatively affected the result of the final ensemble classifier. It would be interesting to measure quantitatively the contribution of each of the 8 neural sub-models in the result of the final classifier. This could then be leveraged to improve the performance of the system.

An additional research direction is the use of linguistic features and metadata. The current model does not explicitly use such features, however Trotzek et al. (2018) showed that they can significantly improve early detection of anorexia.

Lastly, it would be interesting to experiment with more diverse architectures in the neural sub-models (e.g. by using other hidden layer architectures, such as recursive neural networks (Goller and Kuchler, 1996; Socher et al., 2011)) as a way of improving the performance of the current ensemble classifier.

Overall, the aim of this thesis was to make use of current developments in the field of deep learning and apply them to one of the many use cases in the area of natural language processing in clinical psychology. Knowing the benefits of using automatic assessment tools in monitoring the mental health of online users and the rising interest in this field, there is great research potential and many developments yet to come.

Bibliography

Muhammad Abdul-Mageed and Lyle Ungar. 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)*, pages 718–728, Vancouver, Canada, July 2017.

Malak Abdullah and Samira Shaikh. 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)*, pages 350–357, New Orleans, Louisiana, USA, June 2018.

Hessam Amini, Farhood Farahnak, and Leila Kosseim. Natural Language Processing: An Overview. In *Frontiers in Pattern Recognition and Artificial Intelligence*, volume 5, chapter 3, pages 35–55. World Scientific, June 2019.

Chidanand Apté, Fred Damerau, and Sholom M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems (TOIS)*, 12(3):233–251, July 1994.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR 2015)*, San Diego, California, USA, May 2015.

Michael Ball, Rashmi Patel, Richard D. Hayes, Richard JB. Dobson, and Robert Stewart. TextHunter – a user friendly tool for extracting generic concepts from free text in clinical research. In *AMIA Annual Symposium Proceedings*, volume 2014, page 729. American Medical Informatics Association, 2014.

R. E. Banchs. On the construction of more human-like chatbots: Affect and emotion analysis of movie dialogue data. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1364–1367, Kuala Lumpur, Malaysia, December 2017. IEEE.

Angelo Basile, Marc Franco-Salvador, Neha Pawar, Sanja Štajner, Mara Chinea Rios, and Yassine Benajiba. SymantoResearch at SemEval-2019 task 3: Combined neural models for emotion classification in human-chatbot conversations. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval 2019)*, Minneapolis, Minnesota, USA, June 2019.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. DataStories at SemEval-2017 task 4: Deep LSTM with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada, August 2017.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994.

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

Adrian Benton, Margaret Mitchell, and Dirk Hovy. Multitask learning for mental health conditions with limited social media data. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 152–162, Valencia, Spain, April 2017.

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

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, October 2001.

Rafael A Calvo, David N Milne, M Sazzad Hussain, and Helen Christensen. Natural language

processing in mental health applications using non-clinical texts. *Natural Language Engineering*, 23(5):649–685, 2017.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018.

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

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 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, June 2019b.

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

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 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)*, pages 1724–1734, Doha, Qatar, October 2014.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of Gated Recurrent Neural Networks on sequence modeling. In *NIPS 2014 Deep Learning and Representation Learning Workshop*, Montreal, Canada, December 2014.

Andre Cianflone, Yulan Feng, Jad Kabbara, and Jackie Chi Kit Cheung. 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)*, pages 2747–2755, Melbourne, Australia, July 2018.

Glen Coppersmith, Mark Dredze, and Craig Harman. Quantifying mental health signals in twitter. In *Proceedings of the Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality (CLPsych 2014)*, pages 51–60, Baltimore, Maryland, USA, June 2014.

Glen Coppersmith, Mark Dredze, Craig Harman, Kristy Hollingshead, and Margaret Mitchell. CLPsych 2015 shared task: Depression and PTSD on twitter. In *Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality (CLPsych 2015)*, pages 31–39, Denver, Colorado, 2015.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

David Roxbee Cox. *Analysis of binary data*. Routledge, London, UK, 2018.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *Proceedings of the Eleventh International Conference on Web and Social Media*, pages 512–515, Montréal, Canada, May 2017.

Bart Desmet and Véronique Hoste. Emotion detection in suicide notes. *Expert Systems with Applications*, 40(16):6351–6358, 2013.

David DeVault, Kallirroi Georgila, Ron Artstein, Fabrizio Morbini, David Traum, Stefan Scherer, Albert Skip Rizzo, and Louis-Philippe Morency. Verbal indicators of psychological distress in interactive dialogue with a virtual human. In *Proceedings of the Special Interest Group on Discourse and Dialogue Conference (SIGDIAL 2013)*, pages 193–202, Metz, France, August 2013.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019.

Luca Dini and André Bittar. Emotion analysis on Twitter: The hidden challenge. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 3953–3958, Portorož, Slovenia, May 2016.

Eibe Frank and Remco R. Bouckaert. Naive bayes for text classification with unbalanced classes. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2006)*, pages 503–510, Berlin, Germany, September 2006.

Alexander Genkin, David D. Lewis, and David Madigan. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007.

Maria Giatsoglou, Manolis G. Vozalis, Konstantinos Diamantaras, Athena Vakali, George Sarigiannidis, and Konstantinos Ch. Chatzisavvas. Sentiment analysis leveraging emotions and word embeddings. *Expert Systems with Applications*, 69:214–224, 2017.

Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. *Neural Networks*, 1:347–352, 1996.

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

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

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

Chenyang Huang, Amine Trabelsi, and Osmar Zaïane. ANA at SemEval-2019 task 3: Contextual emotion detection in conversations through hierarchical LSTMs and BERT. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval 2019)*, pages 49–53, Minneapolis, Minnesota, USA, June 2019.

Richard G Jackson, Rashmi Patel, Nishamali Jayatilleke, Anna Kolliakou, Michael Ball, Genevieve Gorrell, Angus Roberts, Richard J Dobson, and Robert Stewart. Natural language processing to extract symptoms of severe mental illness from clinical text: the clinical record interactive search comprehensive data extraction (CRIS-CODE) project. *British Medical Journal (BMJ open)*, 7 (1), 2017.

Gilles Jacobs, Els Lefever, and Véronique Hoste. Economic event detection in company-specific news text. In *Proceedings of the First Workshop on Economics and Natural Language Processing*, pages 1–10, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Shengyi Jiang, Guansong Pang, Meiling Wu, and Limin Kuang. An improved k-nearest-neighbor algorithm for text categorization. *Expert Systems with Applications*, 39(1):1503–1509, 2012. ISSN 0957-4174.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain, April 2017. Association for Computational Linguistics.

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

Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng. Some effective techniques for naive bayes text classification. *IEEE Transactions on Knowledge and Data Engineering*, 18(11):1457–1466, November 2006.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR 2015)*, San Diego, California, USA, May 2015.

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

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, pages 319–345. 1999.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

Xihao Liang, Ye Ma, and Mingxing Xu. THU-HCSI at SemEval-2019 task 3: Hierarchical ensemble classification of contextual emotion in conversation. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval 2019)*, pages 345–349, Minneapolis, Minnesota, USA, June 2019.

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

David E Losada and Fabio Crestani. A test collection for research on depression and language use. In *International Conference of the Cross-Language Evaluation Forum for European Languages (CLEF 2016)*, pages 28–39, Evora, Portugal, September 2016.

David E. Losada, Fabio Crestani, and Javier Parapar. eRisk 2017: CLEF lab on early risk prediction on the internet: experimental foundations. In *International Conference of the Cross-Language Evaluation Forum for European Languages (CLEF 2017)*, pages 346–360, Dublin, Ireland, September 2017. Springer.

David E. Losada, Fabio Crestani, and Javier Parapar. Overview of eRisk: Early Risk Prediction on the Internet. In *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum*, pages 343–361, Avignon, France, September 2018.

David E. Losada, Fabio Crestani, and Javier Parapar. Overview of eRisk 2019: Early Risk Prediction on the Internet. In *Working Notes of CLEF 2019 - Conference and Labs of the Evaluation Forum*, Lugano, Switzerland, September 2019.

Veronica Lynn, Alissa Goodman, Kate Niederhoffer, Kate Loveys, Philip Resnik, and H. Andrew Schwartz. CLPsych 2018 shared task: Predicting current and future psychological health from

childhood essays. In *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic (CLPsych 2018)*, pages 37–46, New Orleans, LA, 2018.

Larry M. Manevitz and Malik Yousef. One-class svms for document classification. *Journal of machine Learning research*, 2(Dec):139–154, 2001.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June 2014.

Youness Mansar and Sira Ferradans. Sentence classification for investment rules detection. In *Proceedings of the First Workshop on Economics and Natural Language Processing*, pages 44–48, Melbourne, Australia, July 2018. Association for Computational Linguistics.

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

Matthew Matero, Akash Idnani, Youngseo Son, Sal Giorgi, Huy Vu, Mohammad Zamani, Parth Limbachiya, Sharath Chandra Guntuku, and H. Andrew Schwartz. Suicide risk assessment with multi-level dual-context language and BERT. In *Proceedings of the Sixth Workshop on Computational Linguistics and Clinical Psychology (CLPsych 2019)*, pages 39–44, Minneapolis, Minnesota, June 2019.

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

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

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018.

David N. Milne, Glen Pink, Ben Hachey, and Rafael A. Calvo. CLPsych 2016 shared task: Triaging content in online peer-support forums. In *Proceedings of the Third Workshop on Computational Linguistics and Clinical Psychology (CLPsych 2016)*, pages 118–127, San Diego, CA, USA, June 2016.

Saif Mohammad. #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)*, pages 246–255, Montrèal, Canada, June 2012.

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

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. SemEval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation (SemEval 2018)*, pages 1–17, New Orleans, Louisiana, June 2018.

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

Elham Mohammadi, Hessam Amini, and Leila Kosseim. CLaC lab at SemEval-2019 task 3: Contextual emotion detection using a combination of neural networks and SVM. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval 2019)*, pages 153–158, Minneapolis, Minnesota, USA, June 2019a.

Elham Mohammadi, Hessam Amini, and Leila Kosseim. CLaC at CLPsych 2019: Fusion of neural features and predicted class probabilities for suicide risk assessment based on online posts.

In *Proceedings of the Sixth Workshop on Computational Linguistics and Clinical Psychology (CLPsych 2019)*, pages 34–38, Minneapolis, Minnesota, June 2019b.

Elham Mohammadi, Hessam Amini, and Leila Kosseim. Quick and (maybe not so) easy detection of anorexia in social media posts. In *Working Notes of CLEF 2019 - Conference and Labs of the Evaluation Forum*, Lugano, Switzerland, September 2019c.

Elham Mohammadi, Hessam Amini, and Leila Kosseim. Neural feature extraction for contextual emotion detection. In *Proceedings of the 2019 Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, Varna, Bulgaria, September 2019d.

Bahadorreza Ofoghi, Meghan Mann, and Karin Verspoor. Towards early discovery of salient health threats: A social media emotion classification technique. In *Biocomputing 2016: Proceedings of the Pacific Symposium*, pages 504–515, Kohala Coast, Hawaii, January 2016.

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

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

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. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12 (Oct):2825–2830, 2011.

James W Pennebaker, Martha E Francis, and Roger J Booth. Linguistic inquiry and word count (LIWC2007). 2007.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543, Doha, Qatar, October 2014.

John Pestian, Henry Nasrallah, Pawel Matykiewicz, Aurora Bennett, and Antoon Leenaars. Suicide note classification using natural language processing: A content analysis. *Biomedical informatics insights*, 3:BII–S4706, 2010.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 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)*, pages 2227–2237, New Orleans, Louisiana, USA, June 2018.

Soujanya Poria, Erik Cambria, Newton Howard, Guang-Bin Huang, and Amir Hussain. Fusing audio, visual and textual clues for sentiment analysis from multimodal content. *Neurocomputing*, 174(PA):50–59, January 2016. ISSN 0925-2312.

Victor Ruiz, Lingyun Shi, Wei Quan, Neal Ryan, Candice Biernesser, David Brent, and Rich Tsui. CLPsych2019 shared task: Predicting suicide risk level from Reddit posts on multiple forums. In *Proceedings of the Sixth Workshop on Computational Linguistics and Clinical Psychology (CLPsych 2019)*, pages 162–166, Minneapolis, Minnesota, June 2019.

Guergana K. Savova, James J. Masanz, Philip V. Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C. Kipper-Schuler, and Christopher G. Chute. Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513, 2010.

Judy Hanwen Shen and Frank Rudzicz. Detecting anxiety through reddit. In *Proceedings of the Fourth Workshop on Computational Linguistics and Clinical Psychology – From Linguistic Signal to Clinical Reality (CLPsych 2017)*, pages 58–65, Vancouver, Canada, 2017.

Han-Chin Shing, Suraj Nair, Ayah Zirikly, Meir Friedenberg, Hal Daumé III, and Philip Resnik. Expert, crowdsourced, and machine assessment of suicide risk via online postings. In *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic (CLPsych 2018)*, pages 25–36, New Orleans, Louisiana, USA, June 2018.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Ng, and Chris Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML 2011)*, pages 129–136, Bellevue, Washington, USA, June 2011.

Charles Spearman. The proof and measurement of association between two things. *The American journal of psychology*, 100(3/4):441–471, 1987.

Laura Louise Struik and Neill Bruce Baskerville. The role of facebook in crush the crave, a mobile- and social media-based smoking cessation intervention: qualitative framework analysis of posts. *Journal of medical Internet Research*, 16(7), 2014.

Joseph J. Thompson, Betty H. M. Leung, Mark R. Blair, and Maite Taboada. Sentiment analysis of player chat messaging in the video game StarCraft 2: Extending a lexicon-based model. *Knowledge-Based Systems*, 137:149–162, December 2017.

Marcel Trotzek, Sven Koitka, and Christoph M Friedrich. Word embeddings and linguistic metadata at the CLEF 2018 tasks for early detection of depression and anorexia. In *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum*, Avignon, France, September 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 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)*, pages 5998–6008. Long Beach, California, USA, January 2017.

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

Baoxun Xu, Xiufeng Guo, Yunming Ye, and Jiefeng Cheng. An improved random forest classifier for text categorization. *Journal of Computers*, 7(12):2913–2920, 2012.

Christopher C Yang, Ling Jiang, Haodong Yang, and Xuning Tang. Detecting signals of adverse drug reactions from health consumer contributed content in social media. In *Proceedings of ACM SIGKDD Workshop on Health Informatics (HI-KDD 2012)*, Beijing, China, August 2012.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 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)*, pages 1480–1489, San Diego, California, USA, June 2016.

Jie Yin, Sarvnaz Karimi, Andrew Lampert, Mark Cameron, Bella Robinson, and Robert Power. Using social media to enhance emergency situation awareness. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, Buenos Aires, Argentina, July 2015.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 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)*, pages 207–212, Berlin, Germany, August 2016.

Ayah Zirikly, Philip Resnik, Özlem Uzuner, and Kristy Hollingshead. CLPsych 2019 shared task: Predicting the degree of suicide risk in Reddit posts. In *Proceedings of the Sixth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic (CLPsych 2019)*, pages 24–33, Minneapolis, Minnesota, USA, June 2019.