Automatic Emotion Detection in Text Messages using Supervised Learning

by

Maryam Hasan

A Dissertation

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in

Computer Science

January 15 2021

APPROVED:

Professor Elke A. Rundensteiner
Department of Computer Science
Worcester Polytechnic Institute
Advisor

Professor Emmanuel Agu
Department of Computer Science
Worcester Polytechnic Institute
Committee Member

Professor Kyumin Lee
Department of Computer Science
Worcester Polytechnic Institute
Committee Member

Professor Wei Ding
Department of Computer Science
University of Massachusetts Boston
External Committee Member

Professor Craig Wills
Department of Computer Science
Worcester Polytechnic Institute
Head of Department

Contents

Abstract

Emotion detection from text is the task of detecting affective states from natural language artifacts including comments, reviews, messages, and social media posts. Emotion detection tools could potentially be employed in several fields from social science, political science, public health research to marketing research.

In this dissertation, we study the problem of detecting and analyzing emotion in textual data using traditional machine learning and deep learning methods. Emotion detection entails classifying text into categories of emotions such as happiness, sadness, and anger. Supervised emotion classification is challenging due to the limited number of labeled data resources. Moreover, it is further complicated due to involving a high-dimensional feature space and a large number of emotion categories.

This dissertation designs, develops, and evaluates three innovative strategies. First, we develop Emotex, a supervised emotion classification approach using static feature vectors. Feature extraction is a fundamental building block of emotion classification systems. To solve the problem of the high-dimensional feature space, Emotex relies on hand-crafted features selected from lexicons for deriving word-emotion association. Emotex utilizes embedded hashtags to automatically label the emotions expressed in text messages. It builds a large corpus of emotion-labeled messages with no manual effort to train emotion classifiers. Our experimental results show that Emotex models were able to achieve about 90% accuracy on test data for multi-class emotion classification.

Emotex requires extensive hand-crafted features to achieve high performance due to diverse ways of representing emotions in different domains. Such hand-crafted features are time consuming to create and may be incomplete. To solve this problem, we develop a deep learning approach called DeepEmotex that learns emotion-specific features based on the input textual context instead of using static hand-crafted features. In particular, DeepEmotex learns emotion-specific features using sequential transfer learning. For this, we develop a sequential transfer learning framework to fine-tune the pre-trained language models.

More precisely, DeepEmotex utilizes two state-of-the-art pre-trained models, known as BERT and Universal Sentence Encoder (USE). We analyze the adaptation or fine-tuning phase during which the pre-trained knowledge is transferred to our emotion classification task. We fine-tune our models on a total of 300,000 tweets as our training dataset, validate on 60,000 tweets and use 180,525 tweets as our test dataset. By fine-tuning USE, an overall accuracy of 91% on our test dataset is achieved. Using different batch sizes to fine-tune BERT, we achieve 92% accuracy on our test data. We also evaluate the performance of DeepEmotex models in classifying emotion in EmoInt benchmark dataset. DeepEmotex models obtain state-of-the-art performance on classifying emotion in the benchmark dataset. Evaluation results show that the proposed BERT model outperforms the state-of-the-art result using the Bi-directional-LSTM-CNN model [48] by 3%.

After developing emotion classification models, we deploy the trained models to analyze live streams of tweets. For this, we develop a framework called EmotexStream. First, a binary classifier separates tweets with explicit emotion from tweets without emotion. Then, our emotion classification models are utilized for a fine-grained emotion classification of tweets with explicit emotion. We also propose an online method to measure public emotion and detect abrupt changes in emotion as emotion-burst moments in live text streams. Through a series of case studies we confirm that the proposed methods are able to detect emotion-critical moments during real-life events.

Acknowledgments

I am thankful to many people who made my Ph.D. possible and enjoyable. I would like to express my sincere appreciation to my advisor Professor Elke A. Rundensteiner. I would like to thank Professor Rundensteiner for her patience, motivation, enthusiasm, and immense knowledge guiding me through my Ph.D. program.

My special thank you goes to my dissertation committee members Professor Emmanuel Agu and Professor Kyumin Lee for careful reading of this manuscript and improving it by their thoughtful suggestions. Especially, I am very grateful to Professor Emmanuel Agu, for regularly meeting with me for two years to guide me in the early stage of my research. His attention to detail improved the quality of my research. His valuable feedback on my early research work eventually became my first full papers. I am thankful to Professor Kyumin Lee for joining my dissertation committee and providing feedback on my dissertation. I thank Professor Wei Ding for her time and support as an external committee member.

I would also like to thank all members of the DSRG group for listening to my presentations and enriching them by bright ideas and critical comments. I would like to thank all research collaborators in DSRG. I thank Worcester Polytechnic Institute for supporting me as a teaching assistant during the first three years of my PhD program.

An honorable thank you goes to my family. I thank my parents, for their love and encouragement to pursue my education, even when it goes beyond boundaries of language and geography. I also appreciate my husband, Professor Mojtaba Azadi for his unconditional support in completing this dissertation. Without his support, I would face many difficulties while doing this. A special thank you goes to my dear daughter Alia Azadi for her endless kindness, love, and hugs.

1 Introduction

In recent years, there has been a great deal of interest in automatically identifying opinions, emotions, and sentiments in text. This is due to the rapid growth of social networks, product reviews, discussion forum posts, blogs, microblog posts, and the easy access to a massive subjective and emotional data in the digital format.

What people think and how they feel plays a huge part in their decision making. In particular, emotion plays a critical role in our daily performance affecting many aspects of our lives including social interaction, behavior, attitude, and decision-making [125]. Understanding human emotion patterns and how the people feel play an essential role in various applications including public health and safety, emergency response, and urban planning.

Text is a particularly important source of data for detecting emotion because the bulk of textual data ranging from microblogs, emails, to SMS messages on a smart phone that has become increasingly available. The rapid growth of emotion-rich textual data makes a necessity to automate identification and analysis of people's emotion expressed in text [125].

Social networks and microblogging tools (e.g., Twitter and Facebook) are increasingly used by individuals to share their opinions and feelings in the form of short text messages (e.g., texts about normal life and opinion on current issues and events) [30]. This results in massive amounts of text messages that can be analyzed for a wide range of insights.

Text messages posted on social networks (commonly known as tweets or microblogs) may also contain indicators of emotions of individuals such as happiness, anxiety, and depression. This makes social networks a large corpus of textual data that is rich with emotional content, which can be mined for a variety of purposes. Such networks are appropriate data sources for behavioral studies, especially for studying the emotions of individuals as well as larger populations. Interesting applications of behavioral studies, include detecting mood after a disaster, analyzing political mood, or understanding emotion about certain products. Therefore, social networks such as Twitter provide valuable information to observe crowd emotion and behavior and study a variety of human behavior and characteristics [123].

1.1 Applications of Emotion Detection in Text Messages

Increasing evidence suggests that emotion detection and screening built around social media [25, 89, 41, 105] will be effective in many applications. In particular, Twitter provides valuable opportunities to observe public mood and behavior. The development of robust textual emotion sensing technologies promises to have a substantial impact on public and individual health and urban planning. Such emotion mining tools, once available, could

potentially be employed in a large variety of applications ranging from population level studies of emotions, the provision of mental health counseling services over social media, and other emotion management applications. The census bureau and other polling organizations may be able to use the emotion mining technology to estimate the percentage of people in a community experiencing certain emotions and correlate this with current events and various other aspects of urban living conditions. This type of technology can also enhance early outbreak warning for public health authorities so that a rapid action can take place [56].

The emotion mining tools could also be used by counseling agencies to monitor emotional states of individuals or to recognize anxiety or systemic stressors of populations [43]. For instance, university counseling centers could be warned early about distressed students that may require further personal assessment.

Moreover, studying public emotion promises to be of great value in several fields from social science, political science, public health research to market research, that are interested in aggregate emotion instead of individual cases. It could assist government agencies in recognizing growing public fear or anger associated with a particular decision or event or in helping them to understand the publics emotional response toward controversial issues or international affairs. In some cases rapidly gaining such insights as well as getting a deeper understanding on trends associated with positive versus negative emotion propagation across a population can be critical. The analysis of emotion during real-life events helps to realize the public emotion regarding the event. Important events are often discussed widely in social networks. Public emotion analysis can aid public health researchers by providing them with (1) a low-cost method to detect emotion-critical events across different sub-populations; (2) useful knowledge for identifying at-risk populations; and (3) a method to formulate new hypotheses about the impact of real-time events on populations.

1.2 Challenges of Detecting Emotion in Social Networks

Emotion analysis is a more challenging problem than the binary sentiment classification. While both tasks suffer from the implicit nature of natural language, emotion analysis is further complicated due to the greater number of classification categories (i.e., emotion classes).

Additionally, there is no fixed number or types of emotions, as different models of emotion have been proposed by psychologists. Each model defines a slightly different set of emotions. Categorizing emotion into distinct classes is more difficult not only because emotion detection in general requires deeper insights, but also because of the similarities

between different emotions which make clean classification a challenge. Sometimes emotion classes are even hard for human annotators to distinguish. One notable example in this regard is anger and disgust [7]

The emerging field of multi-class emotion recognition which entails classifying text into several categories of emotion such as happiness, sadness, anger, and more), have remained under-explored due to numerous reasons elaborated in the following.

- *Casual style of microblog data:* Text messages are usually written in a casual style. They may contain numerous grammatical and spelling errors along with abbreviations and slang words. While the use of informal language and short messages has been previously studied in the context of sentiment analysis [37, 87, 13, 60], the use of such language in the context of emotion mining has been much less studied.

- *Data sparsity*: Data sparsity occurs in social networks due to the large amount of informal textual information. The reason of data sparsity in Twitter is the fact that a great percentage of tweet's terms occur fewer than 10 times [109] in the entire corpus. This phenomenon, which causes data sparsity, has an impact on the overall performance of emotion analysis. One study focused on reducing data sparsity of tweets was presented by Saif et al. [109]. They proposed semantic smoothing to reduce data sparsity.

- *Large feature space*: In text classification each word is a feature. Social text streams have generated a large amount of textual data with high features and dimensions. The use of a global vocabulary of millions of words creates a huge feature space and is not efficient.

- *Changes in features*: Relevancy of features may change over time. When a new event or topic is created, new features (words) are appeared that were not considered in the original feature space.

Other than the above challenges related to the problem of text classification, there are also some specific challenges specific to emotion classification as below:

- *Semantic ambiguity of text messages:* Human emotions as well as the texts expressing them are ambiguous and subjective. This makes it difficult to accurately infer and interpret the author's emotional states.

- *Fuzzy boundaries of emotion classes:* Emotions are complex concepts with fuzzy boundaries and with variations in expression. Thus, modeling and analyzing the human affective behavior is a challenge for automated systems [40].

- *Difficulty of emotion annotation:* In order to train an automatic classifier, supervised learning methods require labeled data. It would be time consuming, tedious and labor-intensive to manually label text messages for the purpose of training a classifier for emotion detection.

- *Inconsistent annotators:* While crowdsourcing emotion labels have been explored, human annotators may not be reliable. A human annotator's judgement of the emotions in a text message is likely to be subjective and inconsistent. Consequently, different annotators may classify the same text message into different emotion classes, as confirmed by our user study in Section 3.4.

1.3 Model of Emotion

The emotion models have mainly been studied based on two fundamental approaches: basic emotions model and dimensional model [108].

1.3.1 Basic Emotions Model

According to the *basic emotion model* humans have a small set of basic emotions, which are discrete and detectable by an individual's verbal/nonverbal expression [35]. Researchers have attempted to identify a number of basic emotions which are universal among all people and differ one from another in important ways. A popular example is a cross-cultural study by Paul Ekman *et al.* [35], in which they concluded that six basic emotions are anger, disgust, fear, happiness, sadness, and surprise. Subsequently, many works in the field of emotion detection in texts have been conducted based on this basic emotion model [16, 96, 112, 66]. For example, Bollen *et al.* extracted six dimensions of affect including tension, depression, anger, vigour, fatigue, confusion from Twitter to model public emotion [16].

However, the main drawback of such basic emotion models is that there is no consensus amongst theorists on which human emotions should be included in the basic set of emotions. Moreover, the basic emotions doesn't cover all the variety of emotion expressed by humans in texts. People usually express non-basic, subtle and complex emotions. This problem can't be resolved by using a finer granularity, because the emotions expressed in texts are ambiguous and subjective. For instance, "surprise" as a basic emotion can indicate negative, neutral or positive valence. Also using a finer granularity of emotion makes the distinction of one emotion from another an issue in emotion classification. Therefore, a small number of discrete emotions may not reflect the complexity of the affective states conveyed by humans [108].

1.3.2 Dimensional Model of Emotion

In contrast to the basic emotion model which defines discrete emotions, the *dimensional model* defines emotion on a continuous scale. This model characterizes human emotions by defining their positions along two or three dimensions. Many dimensional models incorporate two fundamental dimensions of emotions namely, valence (i.e., pleasure) and arousal (i.e., activation or stimulation) [108].

The most widely used dimensional model is the Circumplex model of Affect proposed by Russell [107]. As shown in Figure 1, the model suggests that emotions are distributed in a two-dimensional circular space, containing valence and arousal dimensions. Instead of a small number of discrete categories, this model defines the emotion in terms of latent dimensions (e.g., arousal and valence). The horizontal axis presents pleasure and measures how positive or negative a person feels. The vertical axis presents activation and measures if one is likely to take an action. Although the Circumplex model is a well-known model and has long been validated and studied by emotion and cognition theorists, it has rarely been used by computational approaches for automatic emotion analysis in texts [20].

1.4 State-of-the-Art

This section surveys prior works on classifying emotion in texts. Emotion classification methods can be divided into lexicon-based methods and supervised learning methods.

1.4.1 Lexicon-based Classification

lexicon-based approaches relies on labeled dictionaries to calculate the emotional orientation of a text based on the words and phrases that constitute it. Agrawal and An [4] presented an unsupervised context-based methodology that does not depend on affect lexicons. Therefore their model is flexible to classify texts beyond Ekman's model of six basic emotions.

Another unsupervised approach presented by Calvo *et al.* [20] using dimensional emotion model, instead of categorical model. They used a normative database ANEW [18] to produce three-dimensional vectors (valence, arousal, dominance) for each document. They compared this method with different categorical approaches. For the categorical approaches three dimensionality reduction techniques: Latent Semantic Analysis (LSA), Probabilistic Latent Semantic Analysis (PLSA) and Non-negative Matrix Factorization (NMF) were evaluated. Their experiments showed that the categorical model using NMF and the dimensional model tend to perform best.

An important limitation to the lexicon-based method is the small size of available lexical resources, which has an effect on performance at the cost of low recall.

Another disadvantage of lexicons is that they contain direct affective words, that refer explicitly to the emotional states. In contrast, indirect affective words have an implicit connection to the emotional concepts depending on the context that they appear in [112].

1.4.2 Emotion Classification using Machine Learning

The use of lexicons is not the only approach for emotion detection. Many researchers applied supervised learning methods to identify emotion in texts. Supervised classification methods require an emotion-annotated data set and a statistical learning algorithm. Choudhury *et al.* [25] detected depressive disorders by measuring behavioral attributes including social engagement, emotion, language and linguistic styles, ego network, and mentions of antidepressant medication. They utilized these behavioral features to build a binary classifier that can predict whether an individual is vulnerable to depression. They crowdsourced data from Twitter users who have been diagnosed with mental disorders. Similar to our emotion model, they considered four emotional states: positive affect, negative affect, activation, and dominance. They used the LIWC lexicon for computing positive and negative affects and the ANEW lexicon for computing activation and dominance. Their models showed an accuracy of 70% in predicting depression.

Another work by Qadir *et al.* [97] learned a list of emotion hashtags using a bootstrapping framework. They started with a small number of manually defined seed hashtags. For each seed hashtag, they searched Twitter for tweets that contained the hashtag and labeled them with the emotion class of the hashtag. They used these labeled tweets to train supervised emotion classifiers. Then, the emotion classifiers were applied to a large pool of unlabeled tweets to identify candidate emotion hashtags. They collected hashtags for five emotion classes including affection, anger, anxiety, joy and sadness.

Purver *et al.* [96] trained supervised classifiers for emotion detection using automatically labeled Twitter messages. They used the 6 basic emotions identified by Ekman [35] including happiness, sadness, anger, fear, surprise and disgust. They used a collection of Twitter messages, all marked with emoticons or hashtags corresponding to one of six emotion classes, as their labeled data. Their method did better for some emotions (happiness, sadness and anger) than others (fear, surprise and disgust). Their work is similar to ours, however they used categorical emotion model which is different than our dimensional emotion model based on the Circumplex model of affect [107]. Also, their overall accuracy (60%) was lower than the accuracy achieved by our approach.

Another supervised learning work with categorical emotion models was developed by Suttles and Ide [117]. They classified emotion according to a set of eight basic bipolar emotions defined by Plutchick, namely, anger, disgust, fear, happiness, sadness, surprise, trust and anticipation. Similar to our method to collect emotion-labeled data, they used

emoticons and emotion-hashtags to collect a large set of labeled data and produce emotion classifiers.

1.4.3 Emotion Classification using Deep Learning

Recently, text classification task has seen some success in switching from linear models such as support vector machines to non-linear neural-network models. Some researchers used neural network models to learn dense word embeddings. Bengio et al. [15] trained word embeddings using a neural language model together with the model's parameters. Collobert and Weston demonstrated the power of pre-trained word embeddings [26, 27]. They established word embeddings and a neural network architecture that many of today's approaches were built upon. Mikolov et al. [75], brought word embedding to the front through creation of word2vec. Word2vec is a toolkit enabling training and use of pre-trained embeddings. A year later, Pennington et al. [92] introduced GloVe, a competitive set of pre-trained embeddings. They trained word embeddings by building a co-occurrence matrix for a given corpus, which contains how frequently words co-occur together in the corpus. Word2vec is a predictive model, whereas GloVe is a count-based model.

Some researchers studied the effectiveness of neural-network models for sentiment classification in text. Ren. et al. [104] proposed a context based neural network model for Twitter sentiment analysis by incorporating contextualized features from relevant Tweets into the model in the form of word embedding vectors. They showed that significant improvements can be achieved by modeling the context of a given target tweet as a set, using neural pooling functions to extract the most useful features from tweets automatically.

Tang et al. [119] proposed learning sentiment specific word embedding (SSWE), which encodes sentiment information in the continuous representation of words. They developed three neural networks to effectively incorporate the supervision from sentiment polarity of text (e.g. sentences or tweets). They applied SSWE as features in a supervised learning framework for Twitter sentiment classification.

Yoon Kim [57] trained a simple CNN with one layer of convolution on top of word2vec vectors. His results showed that unsupervised pre-training of word vectors is an important ingredient in deep learning for NLP. They also described a simple modification to the architecture to allow the use of both pre-trained and task-specific vectors by having multiple channels.

Poria et al. [95] present a novel way of extracting features from short texts, based on the activation values of an inner layer of a deep convolutional neural network. For this, they train a deep convolutional neural network (CNN) on a training corpus with hand-annotated sentiment polarity labels. However, instead of using it as a classifier, as

Yoon Kim [57] did, they used the values from its hidden layer as features for a much more advanced classifier, which gives superior accuracy. They concatenated the obtained feature vectors and fed the resulting long vector into a supervised classifier.

Severyn and Moschitti [110] also studied sentiment classification using deep learning system. Similar to Yoon Kim [57] their network is composed of a single convolutional layer followed by a non-linearity, max pooling and a soft-max classification layer. They proposed a three-step process to train the parameters of their network for sentiment classification: (i) word embeddings were initialized using a neural language model, which was trained on a large unsupervised collection of tweets; (ii) they used a convolutional neural network to further refine the embeddings on a large distant supervised corpus; (iii) the word embeddings and other parameters of the network obtained at the previous stage are used to initialize the network, which is then trained on a supervised corpus.

Another work for sentiment classification using deep learning system was developed by Zhou et al. [137] using deep feature selection methods in natural language processing. They proposed a novel semi-supervised learning algorithm called active deep network by using restricted Boltzmann machines (RBMs). They applied unsupervised learning based on labeled reviews and abundant unlabeled reviews and then used active learning to identify and select reviews that should be labeled as training data.

Some researchers developed feature selection methods using deep learning models. Li et al. [64] developed a method for selecting input features in a deep neural network for multi-class data. They used elastic net to add a sparse one-to-one linear layer between the input layer and the first hidden layer of a multi-layer perception and select most important features according to their weights in the input layer given after training. They then applied their deep feature selection method to solve the problem of enhancer-promoter interaction using genomics data. The method that they developed using elastic net is a new shrinkage approach that is flexible to be applied in different deep learning architectures. Zou et al. [138] also developed a feature selection model using deep neural network. Their method formulates the feature selection problem as a feature reconstruction problem. It is implemented as an iterative algorithm, which is based on Deep Belief Network in an unsupervised learning way to train the inquired reconstruction weights.

1.5 Research Objectives for Emotion Classification in Text

Our goal is to automatically detect emotion in text messages in social networks. For this, we develop supervised machine learning methods to classify the messages into their emotional states. Supervised learning methods achieve high accuracy in text classification [126]. However they require a set of training records, such that each record is labeled

with a class value. The training data is used to construct a classification model, which relates the features in the underlying input data to one of the class labels. The classification model is used to predict a class label for a test instance.

In order to train a classification model from labeled texts, sentences or messages should be represented as a vector of numerical features. Feature representation is needed to convert text content into a numeric vector representation, which can then be utilized to train a classification model. It is important to know how features are represented in text classification. Thus, to accomplish the emotion classification task, the major challenges discussed below must be tackled.

1.5.1 Collecting Text Data with Emotion Labels

While supervised learning methods achieve high accuracy, they require a large corpus of texts labeled with the emotion classes they express [126]. The training data is used to construct a classification model, which relates the features in the underlying input data to one of the class labels. The classification model is used to predict a class label for a test instance.

Prior works have mostly utilized manually labeled data. Crowdsourcing is a popular approach for labeling data, in which humans manually infer and then annotate each message with the emotion it expresses [25, 30, 89]. Crowdsourcing tools such as Amazon's mechanical turk facilitate access to manual data labelers. However manually labeling of Twitter messages with the emotions they express faces numerous challenges as previously outlined, including the inconsistency of human labelers (See Section 1.2). Therefore, instead we investigate using hashtags (user-selected keywords) in Twitter messages as viable alternative to manual labeling. The use of hashtags in tweets is very common. Twitter contains millions of different user-defined hashtags. Wang *et al.* showed that 14.6% of tweets in a sample of 0.6 million tweets had at least one hashtag [126]. We make the observation that in many cases the hashtag keywords may correspond to the author's own classification of the main topics of their message. A study by Wang *et al.* showed that emotion hashtags in about 93% of their sample tweets are relevant and reflect the writer's emotion [125].

We thus conjecture that emotional hashtags inserted by authors indicate the main emotion expressed by their Twitter message. For example, a tweet with the hashtag "#depressed" can be interpreted as expressing a depressed emotion, while a tweet containing the hashtag "#excited" as expressing excitement. By using embedded hashtags to automatically label the emotions expressed in text messages, we build a large corpus of labeled messages to train classifiers with no manual effort. This approach overcomes the need for manual labeling and yields a completely automatic scheme for labeling a massive

repository of Twitter messages. This strategy could equally be applied in other mining applications where labeling is required.

1.5.2 Selecting Emotion Classes

Another challenge for automated emotion detection is that emotions are complex concepts with fuzzy boundaries and with individual variations in expression and perception. We define the emotion classes based on the Circumplex model of affect [107].

In our emotion classification work, we utilize the Circumplex model of affect by considering four major classes of emotion: Happy-Active, Happy-Inactive, Unhappy-Active, and Unhappy-Inactive. As shown in Figure 1, the defined four classes of emotion are distinct, yet describe a wide range of emotional states as they cover four dimensions of the Circumplex model.

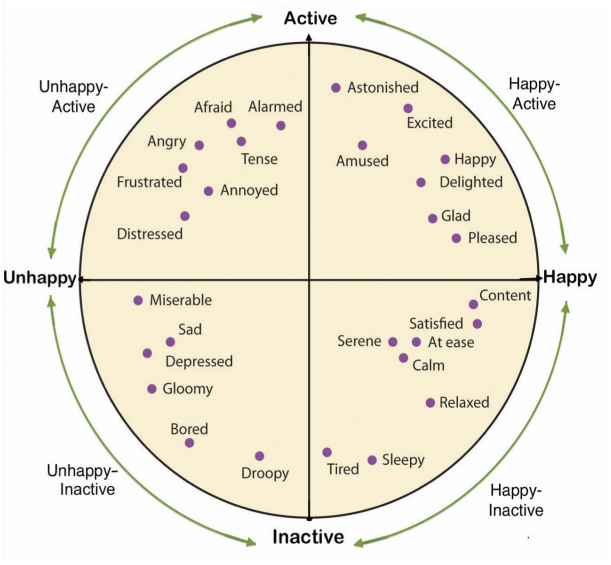

Figure 1: Circumplex model of affect including 28 affect words by J. A. Russell, 1980. [107]

1.5.3 Extracting and Representing Features

Feature representation is essential in natural language processing (NLP) and text classification. It has a significant influence on the performance of many NLP tasks. A word representation is a mathematical object associated with each word, called a vector. This means representing them in a way that computer can understand. Therefore before any NLP task, words or phrases should be represented as a n-dimentional vector of numerical features. Each dimension's value of this vector corresponds to a semantic or grammatical feature of the word [121]. Two different statistical methods for computing the feature

vectors are presented in the following sections.

1.5.3.1 One-Hot Encoding

One-hot encoding creates a feature vector with dimension $1 \times N$ for each word, where N is the size of the vocabulary, and only one dimension is on. For a single word the corresponding column is filled with the value one and the rest are zero valued.

Despite its simplicity, this model usually demonstrate good performance on NLP tasks including sentiment analysis. Many text classification methods use one-hot encoding because of its simplicity and efficiency for classification purposes [3]. However, one-hot representation of a word suffers from several disadvantages [63]. The word vectors created using the one-hot encoding is very large and sparse. The feature vector has the same length as the size of the vocabulary [121]. Considering the large size of training dataset, the number of words in the vocabulary tends to be extremely large. Thus, the feature vector of each word would become excessively large and sparse.

Moreover, it ignores the word orders, and thus different sentences can have exactly the same representation, as long as the same words are used. Another problem is that this model doesn't capture word similarities and semantics. Bag-of-words model has very little sense about the semantics of the words or more formally the distances between the words. This means that words "sad","upset" and "apple" are equally distant despite the fact that semantically, "sad" should be closer to "upset" than "apple" [63].

These limitations of one-hot word representations have prompted researchers to investigate distributed methods for inducing word representations.

1.5.3.2 Distributed Representation

Distributed representation methods exploit word co-occurrences to build dense, low-dimensional and real-valued vectors of words [71]. Distributed representations are dense and low-dimensional vectors, with each factor in the vector representing some distinct informative feature of the word. Each entry in the vector represents a distinct informative property. These representations can capture semantic or syntactic regularities in language [79]. In this model, words with similar meaning can be correspond to close vectors.

Distributed representations (the real values of the vector entries) can be obtained using neural network models [15]. Recent works for learning vector representations of words using neural network models have succeeded in capturing fine-grained semantic and syntactic regularities [15, 26, 121, 75]. They are all based on learning a distributed representation for each word, called a word embedding using neural networks. The word embeddings are typically learned using neural networks as the underlying predictive model [15].

Distributed representations developed in the context of statistical language modeling by Bengio et al. [15] using feed-forward neural network models. Collobert et al. [26] and Turian [121] developed a system where word representations are used with state-of-the-art classifiers to improve performance in many NLP tasks. Mikolov et al. learned distributed word embeddings in the context of recurrent neural network models [76, 77].

1.6 Proposed Approaches for Emotion Analysis in Text Messages

In this dissertation, we study emotion analysis in social networks. To detect and analyze the emotion expressed in text messages, we classify the messages into their emotional states.

For decades, machine learning methods solving NLP problems have been based on linear models (e.g., SVM and logistic regression) trained on very high dimensional, sparse and hand-crafted features [133]. In the last few years, neural networks based on dense vector representations have been producing superior results on various NLP tasks [133]. This trend is sparked by the success of word embeddings [78, 76] and deep learning methods [111]. Deep learning models automatically learn multiple layers of feature representations for NLP tasks and thus reduce the need for hand-crafted features [133].

We propose two learning approaches to classify emotion in text messages: a machine learning approach using static feature vectors and a deep learning approach using distributed feature vectors. We summarize the proposed approaches in the picture 2.

1.6.1 Emotex: A Supervised Learning Approach using Static Feature Vectors

In the first approach we develop a supervised machine learning approach called Emotex to classify emotion expressed in text messages. Emotex collects a large dataset of emotion-labeled messages from Twitter. The messages are pre-processed and converted into feature vectors. Emotex uses the sparse one-hot model to represent each tweet as a feature vector. The feature vector in Emotex is static. The features are selected based on a predefined set of emotion lexicons. Then these vectors are subsequently utilized to train emotion classification models. Emotex utilizes a number of classification methods for text categorization, including Bayesian classifiers, decision trees, and support vector machines (SVM). We select Naive Bayes as a probabilistic classifier, SVM as a decision boundary classifier, and decision tree as a rule based classifier.

1.6.2 DeepEmotex: A Deep Learning Approach using Distributed Feature Vectors

Emotex requires extensive hand-crafted features due to diverse ways of representing emotion in different domains. Such hand-crafted features are time-consuming to create and

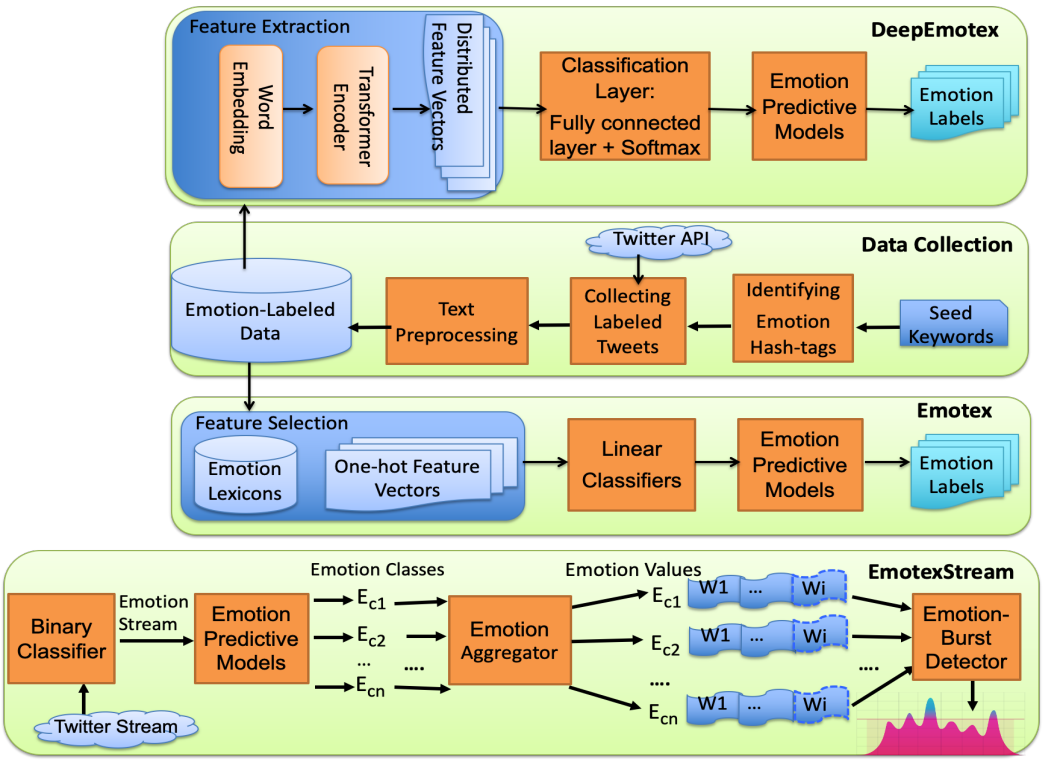

Figure 2: Summary of proposed approaches.

may be incomplete. To solve this problem, we develop a deep learning approach called DeepEmotex to extract distributed dense features based on the input context instead of using static hand-crafted features. DeepEmotex learns emotion-specific features using sequential transfer learning.

Sequential transfer learning consists of two stages: A pre-training phase and an adaptation phase. The pre-training causes the model to learn general-purpose knowledge that can then be transferred to downstream tasks [101]. Often, pre-training objective is to learn general-purpose word or sentence representations [78, 58]. In the adaptation phase, the knowledge of the trained model is transferred to the target task. We utilize state-of-the-art pre-trained neural network models and fine-tune them to do our target emotion classification task.

1.6.3 EmotexStream: A Framework for Analyzing Emotion in Live Streams of Text Messages

After developing emotion classification models, we deploy the trained models to analyze live streams of tweets in a series of case studies. For this application, we develop a two-stage framework called EmotexStream. A binary classifier in the first stage sepa-

rates tweets with explicit emotion from tweets without emotion. The second stage utilizes our emotion classification models for a multi-class emotion classification of tweets with explicit emotion. We propose an online method to measure public emotion and detect abrupt changes in emotion as emotion-intensive moments in live text streams. We then utilize EmotexStream system to measure public emotion and detect emotion-burst moments in live stream of tweets. EmotexStream system is able to detect emotion-critical moments during real-life events in a series of case studies.

We develop DeepEmotex models after EmotexStream system. Thus, DeepEmotex models are not deployed to classify live streams of tweets using EmotexStream framework.

1.7 Contributions

Overall, we made the following contributions in this dissertation:

- Proposed a method to overcome the vague boundaries of emotion classes: We address this issue using a two-pronged approach. First, we define the emotion classes based on the Circumplex model of affect [16]. Instead of a small number of discrete categories, this model defines the emotion in terms of latent dimensions (e.g., arousal and valence). Second, a soft (i.e., fuzzy) classification approach is proposed to measure the probability of assigning a message into each emotion class, in addition to a typical classification that simply assigns one single emotion class to each text message in a deterministic manner.

- Proposed a distant supervision method to collect emotion-labeled data: We conjecture that emotional hashtags inserted by authors indicate the main emotion expressed by their Twitter message. This approach overcomes the need for manual labeling and yields a completely automatic scheme to obtain large amounts of labeled data. This strategy could equally be applied in other applications where labeling is required to automatically obtain large amounts of supervised data.

- Developed and evaluated machine learning models to classify emotion in text: We develop the Emotex system to automatically classify emotion expressed in text messages. Emotex uses sparse one-hot model to represent feature vectors. The features are selected based on a predefined set of emotion lexicons. We train emotion classification models and report their soft and hard classification results. We evaluate the classification accuracy of Emotex by comparing it with the lexical approach. Classification accuracy of Emotex is about 90%, while the accuracy of the lexical approach is about 66%.

- Developed and evaluated neural transfer learning models to classify emotion in text: We develop a deep learning framework called DeepEmotex to classify emotion in text messages. DeepEmotex learns emotion-specific features based on the input textual context using sequential transfer learning. For this, DeepEmotex develops methods for fine-tuning pre-trained language models to learn emotion specific features which are more contextually aware of a new domain. We analyze the adaptation or fine-tuning phase during which the pre-trained knowledge is transferred to our emotion classification task. Using the state-of-the-art pre-trained models, we achieved 92% classification accuracy on our test data. We also evaluate the performance of DeepEmotex models in classifying emotion in the benchmark datasets. DeepEmotex models were able to correctly classify emotion in 70% of the the benchmark dataset.

- Proposed a method to detect emotion-burst moments during real-life events: We develop EmotexSream framework to classify live streams of tweets. We propose an online method to measure public emotion and detect emotion-intensive moments. Our method can be used for real-time emotion tracking during social events. We evaluate EmotexStream framework by conducting several case studies using live and unfiltered streams of tweets during social events.

1.8 Road Map

The dissertation is organized as follows. Section 1 provides the introduction and motivation of this dissertation. Section 2 surveys the recent literature in the fields of text classification and emotion detection in text by highlighting the various supervised learning approaches and deep learning techniques generally employed.

Details of our Emotex system to classify emotion in text messages are illustrated in Section 3. This section describes Emotex framework and its experimental and evaluation results. Section 4 proposes DeepEmotex, our deep learning framework to classify emotion using sequential transfer learning and fine-tuning the pre-trained models for our emotion classification task. This section provides some background knowledge about neural networks, distributed representations, and transfer learning. Then, it describes details about DeepEmotex framework and its experimental and evaluation results. Section 5 illustrates the proposed approach to detect and analyze public emotion in text streams, followed by our case studies.

2 Related Work on Text Classification

This section surveys prior works on classifying emotion in texts. Then, the current research on using Transfer learning for NLP tasks are reviewed.

2.1 Emotion Classification in Text

Emotion detection methods can be divided into lexicon-based, machine learning, and deep learning methods.

2.1.1 Lexicon-based Methods

Most research on textual emotion recognition is based on building and employing emotion lexicons [73, 84, 112]. Lexicon-based methods rely on lexical resources such as lexicons, set of words or ontologies. They usually start with a small set of seed words. Then they bootstrap this set through synonym detection or on-line resources to collect a larger lexicon. Ma *et al.* [73] searched WordNet for emotional words for all 6 emotional types defined by Ekman [35]. They then assigned weights to those words according to the proportion of Synsets with emotional association that the words belong to. Strapparava and Mihalcea [112] constructed a large lexicon annotated for six basic emotions: anger, disgust, fear, joy, sadness and surprise. They used linguistic information from WordNet Affect [113].

In another work, Choudhury *et al.* [30] identified a lexicon of more than 200 moods frequently observed on Twitter. Inspired by the Circumplex model, they measured the valence and arousal of each mood using mechanical turk and psychology literature sources. Then, they collected posts which had at least one of the moods in their mood lexicon as indicated by a hashtag at the end of a post.

Mohammed *et al.* [82] and Wang *et al.* [125] collected emotion-labeled tweets using hashtags for several basic emotions including joy, sadness, anger, fear, and surprise. They showed through experiments that emotion hashtags are relevant and match with the annotations of trained judges. Canales *et al.* also collected emotion-labeled corpora using a bootstrapping process [21]. They annotated sentences from blogs posts based on the Ekman's six basic emotions [35].

Recently, researchers have explored social media such as Twitter to investigate its potential to detect depressive disorders. Park *et al.* [89] ran studies to capture the depressive mood of users in Twitter. They studied 69 individuals to understand how their depressive states are reflected in their tweets. They found that people post about their depression and even their treatments on social media. Their results showed that participants with depression exhibited an increased usage of words related to negative emotions and anger in their

tweets. Another effort for emotion analysis on Twitter data was undertaken by Bollen *et al.* [16]. They extracted 6 basic emotions (tension, depression, anger, vigor, fatigue, confusion) using an extended version of POMS (Profile of Mood States). They found that social, political, cultural and economic events have a significant and immediate effect on the public mood.

2.1.2 Machine Learning Methods

Machine Learning methods apply statistical algorithms on linguistic features, which can be supervised or unsupervised. A few researchers applied supervised learning methods to identify emotions in texts. Choudhury *et al.* [25] detected depressive disorders by measuring behavioral attributes including social engagement, emotion, language and linguistic styles, ego network, and mentions of antidepressant medication. Then they leveraged these behavioral features to build a statistical classifier that estimates the risk of depression. They crowdsourced data from Twitter users who have been diagnosed with mental disorders. Their models showed an accuracy of 70% in predicting depression.

Another work accomplished by Qadir *et al.* [97] to learn lists of emotion hashtags using a bootstrapping framework. Starting with a small number of seed hashtags, they trained emotion classifiers to identify and score candidate emotion hashtags. They collected hashtags for five emotion classes including affection, anger, anxiety, joy and sadness.

Purver *et al.* [96] tried to train supervised classifiers for emotion detection in Twitter messages using automatically labeled data. They used the 6 basic emotions identified by Ekman [35] including happiness, sadness, anger, fear, surprise and disgust. They used a collection of Twitter messages, all marked with emoticons or hashtags corresponding to one of six emotion classes, as their labeled data. Their method did better for some emotions (happiness, sadness and anger) than others (fear, surprise and disgust). Their work is similar to ours, however they used categorical emotion models and their overall accuracies (60%) were much lower than the accuracy achieved by our approach.

Another supervised learning work with categorical emotion models is done by Suttles and Ide [117]. They classify emotions according to a set of eight basic bipolar emotions defined by Plutchick including anger, disgust, fear, happiness, sadness, surprise, trust and anticipation. This allows them to treat the multi-class problem of emotion classification as a binary problem for opposing emotion pairs.

An unsupervised method was proposed by Agrawal and An [4]. They presented an unsupervised context-based approach based on a methodology that does not depend on any existing affect lexicon, therefore their model is flexible to classify texts beyond Ekman's model of six basic emotions. Another unsupervised approach was developed by Calvo *et al.* [20]. They proposed an unsupervised method using dimensional emotion

model. They used a normative database ANEW [18] to produce tree-dimensional vectors (valence, arousal, dominance) for each document. They also compared this method with different categorical approaches. For the categorical approaches three dimensionality reduction techniques: Latent Semantic Analysis (LSA), Probabilistic Latent Semantic Analysis (PLSA) and Non-negative Matrix Factorization (NMF) were evaluated. Their experiments showed that the categorical model using NMF and the dimensional model tend to perform best.

2.1.3 Deep Learning Methods

Recently, approaches which employ deep learning methods for emotion and sentiment detection in text have been proposed. They use word embeddings as input, which already encode some semantic and syntactic information. Lai et al. [61] proposed recurrent CNNs to capture contextual information as far as possible when learning word representations. They also employ a max-pooling layer that automatically explores the words with key roles in text classification to capture the key components in texts. Their model outperforms CNN and Recursive neural networks based on four different text classification datasets.

Ren. et al. [104] proposed a context based neural network model for Twitter sentiment analysis by incorporating contextualized features from relevant Tweets into the model in the form of word embedding vectors. They showed that significant improvements can be achieved by modeling the context of a given target tweet as a set, using neural pooling functions to extract the most useful features from tweets automatically. Another context-sensitive method for sentiment classification proposed by Teng et al. [120]. Their method is based on a simple weighted-sum model, using bidirectional LSTM to learn the sentiment strength, intensification and negation of lexicon sentiments in composing the sentiment value of a sentence.

Qian et al. [98] presented a linguistically regularized LSTM for the task. The proposed model incorporates linguistic resources such as sentiment lexicon, negation words and intensity words into the LSTM in order to capture the sentiment effect in sentences more accurately. Wang et al. [124] combined CNN and LSTM. They proposed a regional CNN-LSTM model, which consists of two parts: regional CNN and LSTM, to predict the valence and arousal ratings of text.

Felbo et al. [36] used millions of emoji occurrences in social networks as noisy labels for pre-training neural models in order to learn richer representations of emotional contexts. To capture the context of each word they use two bidirectional LSTM layers with 1024 hidden units (512 in each direction), with an attention layer that takes all of LSTM layers as input using skip-connections. Through emoji prediction on a dataset of 1.2 billion tweets containing one of 64 common emojis they obtain state-of-the-art performance

for sentiment, emotion and sarcasm detection using a single pre-trained model. Their results confirm that the distant supervision to a more diverse set of noisy labels enables the models to learn better representations of emotional content in text and obtain better performance for detecting sentiment, emotions and sarcasm.

Abdul-Mageed and Ungar [1] build a dataset of tweets including fine-grained emotions. Then they develop Gated Recurrent Neural Networks (GRNNs) across 24 fine-grained types of emotions. Koper et al. [59] predict emotion intensity in tweets by applying deep learning method with extended lexicons of affective norms. They show that domain-specific embeddings (trained on twitter data) perform superior to other embeddings.

Some researchers predict emotion in textual dialogues. Luo and Wang [72] fine-tune BERT model to predict emotion in dialogues by choosing from four emotion classes, joy, sadness, anger, and neutral. They use datasets consisting of scripts from the TV show, Friends, and the anonymous Facebook chat logs named EmotionPush. Chatterjee et al. [23] use BiLSTM model to infer the underlying emotion from textual dialogues by choosing from four emotion classes, happy, sad, angry, and other. Another method to classify emotion in dialogues is developed by Al-Omari et al. [5]. They classify EmoContext dataset into happy, sad, angry and other. They use GloVe embeddings and features extracted from AffectiveTweets. They also extract word contextual embeddings from BERT model. These vectors feed feed-forward and BiLSTM models to obtain predictions. Their result show that the performance of the system is increased by extracting BERT embeddings then feeding them into an BiLSTM network.

2.2 Transfer Learning for NLP Tasks

Transfer learning, where a model is first pre-trained on a data-rich task before being fine-tuned on a downstream task, has emerged as a powerful technique in natural language processing (NLP). Recent advances in transfer learning for NLP have come from a wide variety of developments, such as new pre-training objectives, model architectures, and fine-tuning methods. A survey of Some pre-training objectives and fine-tuning methods are provided in this section. They have produced a wide landscape of pre-training objectives [52, 31, 131, 34], and fine-tuning methods [52, 93, 51].

2.2.1 Pre-training Methods

Recently, it has become increasingly common to pre-train the entire model on a data-rich task. The main utility of transfer learning is the possibility of leveraging pre-trained models. Ideally, this pre-training causes the model to develop general-purpose abilities and

knowledge that can then be transferred to downstream tasks [101].

Modern techniques for transfer learning on NLP tasks often pre-train a language model using unsupervised learning of neural networks on a diverse corpus of unlabeled text. They are then followed by discriminative fine-tuning on each specific task. This approach has recently been used to obtain state-of-the-art results in many of the most common NLP benchmarks [31, 52, 94, 131, 68, 62, 34, 99, 100].

Beyond its empirical strength, unsupervised pre-training for NLP is particularly attractive because unlabeled text data is available enormously. Neural networks have been shown remarkable scalability, i.e. they often achieve better performance simply by training a larger model on a larger dataset [101]. This synergy has resulted in a great deal of recent work developing transfer learning methodology for NLP tasks, which has produced a wide landscape of pre-training objectives [52, 31, 131, 34].

Dai et al. [29] pre-trained a neural network to fine-tune a language model. Through pre-training phase they captured some linguistic information using LSTM models. With pre-training, they were able to achieve strong performance in text classification tasks. The weights obtained from pre-training can then be used as an initialization for the standard LSTMs. They found that a simple pre-training step can significantly stabilize the training of LSTMs. However their model requires millions of in-domain documents to achieve good performance, which severely limits its applicability.

Howard and Ruder [52] proposed Universal Language Model Fine-tuning (ULMFiT), a transfer learning method that can be applied to NLP tasks. ULMFiT can be used to achieve transfer learning for NLP tasks. They introduced techniques for fine-tuning a language model by gradual unfreezing to retain previous knowledge and avoid catastrophic forgetting during fine-tuning.

Another work by Radford et al. [99] pre-trained a neural network using a language modeling objective and then fine-tuned it on a target task with supervision. They made use of task-aware input transformations during fine-tuning to achieve effective transfer while requiring minimal changes to the model architecture. They chose transformer networks to capture longer range linguistic structure. They demonstrated the effectiveness of their model on a wider range of tasks including natural language inference, paraphrase detection and story completion.

Peters et al. [94] introduced a new type of deep contextualized word representation called ELMo (Embeddings from Language Models). ELMo vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus. They used hidden representations from a pre-trained language as auxiliary features while training a supervised model on the target task. This involves a substantial amount of new parameters for each separate target task.

More recently, Raffel et al. [101] pre-trained a standard Transformer using a simple denoising objective and then separately fine-tuned on each of their downstream tasks. we require minimal changes to our model architecture during transfer. Instead of fine-tuning all of the models parameters, they focus on two alternative fine-tuning approaches that update only a subset of the parameters of their model.

BERT (Bidirectional Encoder Representations from Transformers) [31] is another pre-trained model that has achieved state-of-the-art performance in many NLP tasks. BERT is one of the key innovations in the recent progress of contextualized representation learning [31, 52, 94, 99]. The idea behind the progress is that even though the word embedding [92, 78] layer (in a typical neural network for NLP) is trained from large-scale corpora, training a wide variety of neural architectures that encode contextual representations only from the limited supervised data on end tasks is insufficient. Unlike ELMo [94] and ULMFiT [52] that are intended to provide additional features for a particular architecture that bears humans understanding of the end task, BERT adopts a fine-tuning approach that requires almost no specific architecture for each end task. This is desired as an intelligent agent should minimize the use of prior human knowledge in the model design. Instead, it should learn such knowledge from data [129].

RoBERTa [68] showed that the performance of BERT can further improved by small adaptations to the pre-training process. While RoBERTa was able to improve the performance for several supervised tasks, there is no significant difference between BERT and RoBERTa for generating sentence embeddings [103].

2.2.2 Fine-tuning Methods

It is impractical to train language models from scratch with limited computational resources. Instead, it is practical to adapt language models pre-trained from formal texts to another domains.

One approach is the setting where all parameters of a model are pre-trained on an unsupervised task before being fine-tuned on individual supervised tasks. While this approach is straightforward, various alternative methods for training the model on downstream/-supervised tasks have been proposed. In this section, we compare different schemes for fine-tuning the model [101].

It has been argued that fine-tuning all of the models parameters can lead to suboptimal results, particularly on low-resource tasks [93]. Early results on transfer learning for text classification tasks advocated fine-tuning only the parameters of a small classifier that was fed sentence embeddings produced by a fixed pre-trained model [114, 28, 69, 58]. Alternative fine-tuning approaches update only a subset of the parameters.

As an alternative, some researchers proposed transfer with adapter modules. Adapter

modules yield a compact and extensible model. They add only a few trainable parameters per task, and new tasks can be added without revisiting previous ones. The parameters of the original network remain fixed, yielding a high degree of parameter sharing [51]. The idea of "Adapter layers" [51], is motivated by the goal of keeping most of the original model fixed while fine-tuning. Adapter layers are additional dense-ReLU-dense blocks that are added after each of the preexisting feed-forward networks in each block of the Transformer. These new feed-forward networks are designed so that their output dimensionality matches their input. This allows them to be inserted into the network with no additional changes to the structure or parameters. When fine-tuning, only the adapter layer and layer normalization parameters are updated. The main hyper-parameter of this approach is the inner dimensionality of the feed-forward network, which changes the number of new parameters added to the model [101].

The second alternative fine-tuning method is "gradual unfreezing" [52]. In gradual unfreezing, more and more of the model's parameters are fine-tuned over time. Gradual unfreezing was originally applied to a language model architecture consisting of a single stack of layers. In this setting, at the start of fine-tuning only the parameters of the final layer are updated, then after training for a certain number of updates the parameters of the second-to-last layer are also included, and so on until the entire network's parameters are being fine-tuned [101].

3 Emotex: A Supervised Learning Approach to Detect Emotion in Text Messages

3.1 Introduction

Social networks and microblogging tools (e.g., Twitter, Facebook) are increasingly used by individuals. Using these platforms, users express their opinions and share their feelings in the form of short text messages (e.g., texts about normal life and opinion on current issues and events) [30]. These messages (commonly known as tweets or microblogs) may also contain indicators of emotions of individuals such as happiness, anxiety, and depression. In fact, social networks contain a large corpus of public real-time data that is rich with emotional content. This makes them appropriate data sources for behavioral studies, especially for studying emotions of individuals as well as larger populations. Therefore, social networks such as Twitter provide valuable information to observe crowd emotion and behavior and study a variety of human behavior and characteristics [123]. Methods to analyze these data provide an opportunity to conveniently and deeply explore and understand human behavior and emotion patterns.

Increasing evidence suggests that emotion detection and screening built around social media [25, 89, 41, 105] will be effective in many applications. In particular, Twitter provides valuable opportunities to observe public mood and behavior. The development of robust textual emotion sensing technologies promises to have a substantial impact on public and individual health and urban planning. To detect and analyze the emotion expressed in text messages, we develop a machine learning approach to automatically classify the messages into their emotional states.

The rest of this chapter is organized as follows. Details of our proposed methods to detect and analyze emotion in text streams are illustrated in Section 3.2. Section 3.3 includes our extensive experimental results about different tasks of our approach. Evaluating our labeling method is described in Section 3.4. Finally we conclude this chapter in Section 3.5.

3.2 Emotex: Proposed Approach to Classify Emotion in Text Messages

To detect and analyze the emotion expressed in text messages, we develop a supervised learning approach to automatically classify the messages into their emotional states.

While supervised learning methods achieve high accuracy, they require a large corpus of texts labeled with the emotion classes they express [126]. Prior works have mostly utilized manually labeled data. Crowdsourcing is a popular approach for labeling data, in which humans manually infer and then annotate each message with the emotion it expresses [25, 30, 89]. Crowdsourcing tools such as Amazon's mechanical turk facilitate

access to manual data labelers. However manually labeling of Twitter messages with the emotions they express faces numerous challenges as previously outlined, including the inconsistency of human labelers (See Section 3.1). Therefore, instead we investigate using hashtags (user-selected keywords) in Twitter messages as viable alternative to manual labeling. The use of hashtags in tweets is very common. Twitter contains millions of different user-defined hashtags. Wang *et al.* showed that 14.6% of tweets in a sample of 0.6 million tweets had at least one hashtag [126]. We make the observation that in many cases the hashtag keywords may correspond to the author's own classification of the main topics of their message. A study by Wang *et al.* showed that emotion hashtags in about 93% of their sample tweets are relevant and reflect the writer's emotion [125].

We thus conjecture that emotional hashtags inserted by authors indicate the main emotion expressed by their Twitter message. For example, a tweet with the hashtag "#depressed" can be interpreted as expressing a depressed emotion, while a tweet containing the hashtag "#excited" as expressing excitement. By using embedded hashtags to automatically label the emotions expressed in text messages, we build a large corpus of emotion-labeled messages to train classifiers with no manual effort. This approach overcomes the need for manual labeling and yields a completely automatic scheme for labeling a massive repository of Twitter messages. This strategy could equally be applied in other mining applications where labeling is required.

Another challenge for automated emotion detection is that emotions are complex concepts with fuzzy boundaries and with individual variations in expression and perception. We address this issue using a two-pronged approach. First, we define the emotion classes based on the Circumplex model of affect [107]. Instead of a small number of discrete categories, this model defines the emotion in terms of latent dimensions (e.g., arousal and valence). We utilize the Circumplex model by considering four major classes of emotion: Happy-Active, Happy-Inactive, Unhappy-Active, and Unhappy-Inactive.

Second, a soft (i.e., fuzzy) classification approach is proposed, which classifies each message into multiple emotion classes with different probabilities (i.e., weights), instead of forcing each message to be in one emotion class only.

An important challenge for emotion classification in texts is feature selection and word representation. A word representation is a mathematical object associated with each word, called a vector. Therefore before any classification task, words or phrases should be represented as a n-dimensional vector of numerical features. Several statistical methods have been developed for word or phrase representation. Emotex uses one-hot encoding and creates a vector for each word with filled zeros, except one dimension. Despite its simplicity, this model usually demonstrate good performance on classification tasks including sentiment analysis [3]. However, the word vectors created using the one-hot encoding is

very large and sparse. The feature vector has the same length as the size of the vocabulary [121]. To solve the problem of high-dimensional feature space, we select an emotion lexicon as the set of input vocabulary, instead of all the words in the input dataset.

By utilizing the above ideas, we develop a supervised learning system called Emotex to automatically classify text messages into our defined classes of emotion described in Section 1.3.2. Emotex is developed as an offline system and includes three parts. The first part involves data acquisition and collecting training data. The second part is related to feature representation and the third part creates the emotion classifiers. Figure 3 shows the process flow of Emotex. Emotex collects a large dataset of emotion-labeled messages from Twitter. The messages are then preprocessed and converted into feature vectors. We select certain features and represent each tweet in the training set as a numerical feature vector using one-hot model. The numerical feature vectors annotated with emotion labels are used to train emotion classification models. These models then classify unlabeled messages into an appropriate emotion class.

Emotex classification algorithms will receive a sample of training points from our labeled dataset $\mathcal{D}$, which we will denote by

$$\mathcal{D} = (t_1, e_1), ..., (t_n, e_n), \;\; t_i \in \mathcal{T}, \;\; e_i \in E_{class} \tag{1}$$

where $\mathcal{T}$ is the set of all tweets in the labeled dataset $\mathcal{D}$, and E_{class} is the set of emotion labels. Based on the Circumplex model of emotion [107] we defined E_{class} as below:

$E_{class} = \{happy\_active, happy\_inactive, unhappy\_active, unhappy\_inactive\}$

Our emotion classifier is a function that maps a sample tweet t from our test dataset to an emotion class e.

$$e = f(t), \;\; t \in \mathcal{T}, \;\; e \in E_{class} \tag{2}$$

This section now describes each part of the Emotex pipeline.

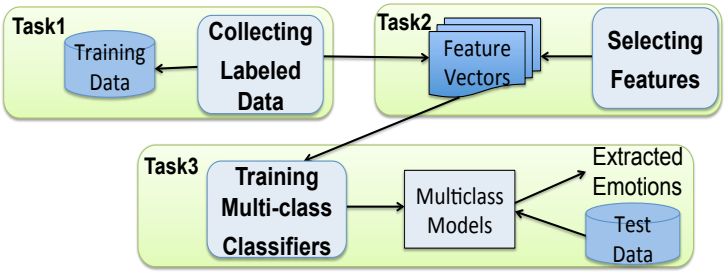

Figure 3: Model of Emotex

Class	Emotion Hashtags
Happy-Active	#elated,#overjoyed,#enjoy,#joyful,#excellent,#feelhappy,#happy,#sohappy, #veryhappy,#happytweet#superhappy,#blessed,#feelblessed,#amazing, #wonderful,#excited,#proud,#enthusiastic,#delighted **Sample: Thankful for unexpected time with one of my best friends #happy**
Happy-Inactive	#calm,#calming,#peaceful,#quiet,#silent,#serene,#convinced,#consent, #contented,#contentment,#satisfied,#relax,#relaxed,#relaxing,#sleepy, #sleepyhead,#asleep, #resting, #restful,#placid **Sample: ready for a relaxing day of doing nothing #relax**
Unhappy-Active	#nervous,#anxious,#tension,#afraid,#fearful,#angry,#stressed,#stress, #distressed,#distress,#stressful,#annoyed,#annoying, #worried,#tense, #bothered,#disturbed, #irritated,#furious, #mad **Sample: I have my speech in less than minutes #nervous**
Unhappy-Inactive	#sad,#ifeelsad,#feelsad,#sosad,#verysad,#sorrow,#supersad,#disappointed, #miserable, #hopeless, #depress,#depressed, #depression,#fatigued,#gloomy, #suicidal,#downhearted, #hapless,#dispirited **Sample: Sometimes people let you down and it hurts. #sad**

Table 1: List of hashtags for each emotion class

3.2.1 Collecting Labeled Data

An emotion-labeled corpus and an emotion lexicon are the two necessary resources for our Emotex system. We utilize hashtags to automatically annotate text messages with emotion and collect a large corpus of emotion-labeled messages. These messages then serve as a labeled dataset for training classifiers. Figure 4 shows the steps of collecting labeled data. We first need to define a list of emotion hashtags to collect emotion-labeled messages. For this, we exploit the set of 28 affect words from the Circumplex model (as shown in Figure 1) as the initial set of keywords and extend them using WordNet's synsets [**?**]. We use the extended set of keywords to detect emotion hashtags. Then, we collect tweets which contain one or more hashtags that fall in our defined list of emotion hashtags. This way we assure that we have tweets labeled with our defined emotion classes described in Section 1.3.2. Hashtags that are directly interleaved in the actual tweet text are more likely to represent a part of the content of the tweet itself [30, 125]. Therefore, we only collect the tweets which contain the emotion hashtags at the end. We also didn't collect retweets, which begin with the"RT" keyword. Table 1 presents the final list of hashtags used to collect labeled data for each class.

Using this approach we are able to collect a large number of tweets with various emotion hashtags with no manual effort. Another major advantage of this approach is that it gives us direct access to the author's own intended emotional state, instead of relying on the possibly inconsistent and inaccurate interpretations of third-party annotators about

what an author of a tweet may have felt. We utilize Twitter's stream API to automatically collect tweets and filter them by emotion-hashtags. After collecting the same number of tweets for each emotion class, the labeled tweets are then preprocessed to mitigate misspellings and casual language used in Twitter using the following rules:

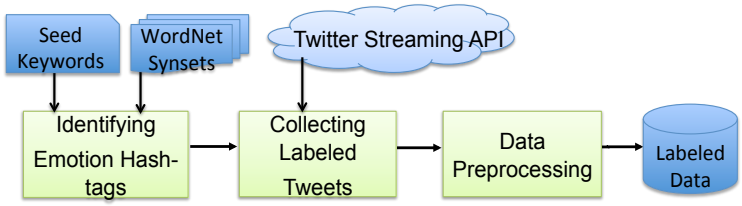

Figure 4: Model of labeled data collection

- *User IDs and URLs:* In addition to the message body, tweets contain the ID of the user and URL links. They are marked separately for later processing.
- *Text normalization:* Tweets often contain abbreviations and informal expressions. All abbreviations are expanded (e.g., "won't" to "will not"). Words with repeated letters are common. Any letter occurring more than two times consecutively is replaced with one occurrence. For instance, the word "happyyyy" will be changed into "happy".
- *Conflicting hashtags:* Some tweets may contain hashtags from different emotion classes. For example tweet "Got a job interview with At&t... #nervous #happy.", includes the hashtag #nervous from Unhappy-Active class and the tag #happy from Happy-Active class. Tweets with conflicting hashtags are removed from our labeled data, as they illustrate a mixture of different emotions.
- *Hashtags at end of tweets:* We consider emotion hashtags at the end of the tweets as emotion labels. Therefore, as part of preprocessing, emotion hashtags are stripped off from the end of tweets. For instance, the tags "#disappointed" and "#sad" are removed from the tweet "No one wants to turn up today. #disappointed #sad". Hashtags that are directly interleaved in the actual tweet text represent part of the content of the tweet and are not removed.

3.2.2 Word Representation for Emotion Classification

Word representation is essential in text classification. It has a significant influence on the performance of classification. A word representation is a mathematical object associated with each word, called a vector. This means representing them in a way that computer

can understand. Therefore before any classification task, words or phrases should be represented as a n-dimentional vector of numerical features. Each dimension's value of this vector corresponds to a semantic or grammatical feature of the word [121] Different statistical methods for computing the word vectors have been developed including one-hot vectors and dense vectors. Emotex system uses One-Hot Encoding to represent each tweet as a numerical vector. Using this model, We represent each word as a completely independent token.

3.2.2.1 One-Hot Encoding

One-hot vector represent every word as an $\mathbb{R}^{|V|\times 1}$ vector with all 0s and one 1 at the index of that word in the vocabulary. In this notation $|V|$ is the size of the vocabulary. For a single word the corresponding column is filled with the value one and the rest are zero valued.

The occurrence (or frequency, or TF-IDF) of each word in text is used as a numeric value which is then transformed into a feature vector using a one-hot representation. Each word is associated with its own dimension. Each dimension in the resulted feature vector represents a feature. Dimensionality of the vector is same as number of distinct features. Features are completely independent from one another. Such a representation is essentially independent of the sequence of words in the texts. Some example word vectors in this type of encoding would appear as the following:

$$W^{a} = \begin{pmatrix} 1 \\ 0 \\ ... \\ 0 \end{pmatrix}, W^{at} = \begin{pmatrix} 0 \\ 1 \\ ... \\ 0 \end{pmatrix}, ..., W^{zebra} = \begin{pmatrix} 0 \\ 0 \\ ... \\ 1 \end{pmatrix}$$

Despite its simplicity, this model usually demonstrate good performance on text classification tasks including sentiment and emotion analysis. Many text classification methods use one-hot encoding because of its simplicity and efficiency for classification purposes [3]. However, one-hot representation of a word suffers from several disadvantages [63]. Text is a particular kind of data in which the word attributes are sparse and high dimensional [3]. The word vectors created using the one-hot encoding are very large and sparse. The word vectors have the same length as the size of the vocabulary [121]. Considering the large size of training dataset, the number of words in the vocabulary tends to be extremely large. Thus, the numerical vector of each word would become excessively large and sparse.

Another problem of one-hot encoding is that this model doesn't capture word similarities and semantics. Bag-of-words model has very little sense about the semantics of the words or more formally the distances between the words. This means that words

"sad","upset" and "apple" are equally distant despite the fact that semantically, "sad" should be closer to "upset" than "apple" [63].

To overcome the problem of this high-dimensional feature space, we select an emotion lexicon as the set of unigram features. As a result, our feature space only contains the emotional words from the emotion lexicons instead of all the words in our training dataset. This method reduces the size of feature space dramatically, with minimal loss of informative terms.

3.2.3 Feature Selection for Capturing Emotion

In order to train a classifier from labeled data, messages or tweets should be represented as a vector of numerical features. Feature selection and text representation are primary tasks for text classification. Feature selection and representation are especially important in text classification due to the high dimensional of text features and the existence of irrelevant and noisy features. In general, text can be represented in different ways as described below [3].

We represent each tweet as a vector of numerical features using one-hot encoding. This vector encodes features such as words, part-of-speech tags or other linguistic and semantic information of the input text. Thus, a set of features that illustrate the emotion expressed by each tweet is needed. We investigate the effectiveness of different features. We use single words, also known as unigrams, as the baseline features for comparison. Other features explored include emoticons, punctuations, and negations.

3.2.3.1 Unigram Features

Unigrams or single word features have been widely used to capture sentiment or emotion in text [37, 87, 96]. Let $\{f_1, f_2, ..., f_m\}$ be our predefined set of unigrams that can appear in a tweet. Each feature f_i in this vector is a word occurring in the list of tweets in our dataset. However, with the large breadth of topics discussed in microblogs, the number of words in our input dataset tends to be extremely large. Thus, the one-hot feature vector of each message would become excessively large and sparse (i.e., most features will have a value of zero). To overcome the problem of this high-dimensional feature space, we select an emotion lexicon as the set of unigram features. As a result, our feature space only contains the emotional words from the emotion lexicons instead of all the words in our training dataset. This method reduces the size of feature space dramatically, with minimal loss of informative terms.

We use different emotion lexicons in our system, including ANEW lexicon (Affective Norms for English Words) [18], LIWC dictionary (Linguistic Inquiry and Word Count) [91],

and AFINN [85]. LIWC is a dictionary of several thousands words and prefixes, grouped into psychological categories. We use emotion-indicative categories including positive emotions, negative emotions, anxiety, anger, sadness, and negations. ANEW lexicon contains 2477 affect words, each rated for its valence and arousal on a 1-9 scale. AFINN was created to include a new word list specifically for microblogs.

3.2.3.2 Emoticon Features

Other than unigrams, emoticons are also likely to be useful features to classify emotion in texts as they are textual portrayals of emotion in the form of icons. Emoticons tend to be widely used in sentiment analysis. Go *et al.* and Pak *et al.* [37, 87] used the western-style emoticons to collect labeled data. There are many emoticons to express happy, sad, angry or sleepy emotion. The list of emoticons that we use can be found in our paper [44].

3.2.3.3 Punctuation Features

Other features potentially helpful for emotion detection are punctuations (i.e., question mark, exclamation mark and combination of them). Users often use exclamation marks when they want to express their strong feelings. For instance, the tweet "I lost 4lb in 3 days!!" expresses strong happiness and the tweet "we're in december, which means one month until EXAMS!!!" represents a high level of stress. The exclamation mark is sometimes used in conjunction with the question mark, which in many cases appears to convey a sense of astonishment. For example the tweet "You don't even offer high speed, yet you keep overcharging me?!" indicates an astonished and annoyed feeling.

3.2.3.4 Negation Features

As our last feature, we select negation to address errors caused by tweets that contain negated phrases like "not sad" or "not happy". For example the tweet, "I'm not happy about this trade." should not be classified as a happy tweet, even though it has a happy unigram. To tackle this problem we define negation as a separate feature. We select the list of phrases indicating negation from the LIWC dictionary.

3.2.4 Classifier Selection for Emotion Detection

A number of classification methods have been commonly used for text classification, including Bayesian classifiers, decision trees, support vector machines (SVM), and Neural Networks. To classify emotion we explored three different classifiers. We selected Naive

Bayes as a probabilistic classifier, SVM as a decision boundary classifier, and decision tree as a rule based classifier.

Using the training set a classifier function $f : (X \rightarrow y)$ is developed that assigns a class label y to an input feature vector X and is used to classify future unlabeled instances. The input vector X encodes features such as words, part-of-speech tags or other linguistic and semantic information of the input text. The output vector y is the emotion class.

One of the challenges of automated emotion detection is that emotions are complex concepts with fuzzy boundaries and with many variations in expression. Also, emotion perception is naturally subjective. Thus, it is difficult to achieve a common consensus to which emotion class each text message belongs to. As shown in our user studies described in Section 3.4, people often have different perceptions about emotion expressed in texts. Furthermore, a small number of discrete emotion classes may not reflect the complexity of the emotional states conveyed by humans. Typical classifiers assume clearly demarcated and non-overlapping classes. They may not assign emotion labels to some messages with high confidence and classify them either incorrectly or correct mostly by chance. Therefore, simply assigning one single emotion class to each text message in a deterministic manner may not perform well in practice.

To overcome this issue we use a two-pronged approach. First, we define the emotion classes based on a dimensional model (See Section 1.3.2). Second, a soft (fuzzy) classification approach is proposed to measure the strength of each emotion class in association with the message under classification. In soft classification, the prediction results become less explicit by assigning each message a soft label that indicates how likely each emotion would be perceived. More details about hard and soft classification of emotion are described bellow.

3.2.4.1 Hard and Soft Classification of Emotion

For classifying emotion, we utilize two types of classification: soft and hard classification. In general, a classifier is a function that assigns an emotion label y to an input feature vector x:

$$y = f(x), \;\; x \in X, \;\; y \in Y \tag{3}$$

where X is the set of all feature vectors from the tweets in the input dataset, and Y is the set of emotion labels.

Some classifiers such as support vector machines make decision boundaries between different classes. Other classifiers are probabilistic classifiers meaning that they assign a probability distribution over a set of classes to an input $x \in X$.

$$P(Y = y|x), \;\; x \in X, \;\; y \in Y \tag{4}$$

In hard classification each message can only belong to one and only one class. Soft classifiers measure the degree to which a message belongs to each class, rather than dedicating the message to a specific class [**?**]. In decision boundary classifiers, soft labels can be estimated based on decision scores. In probabilistic classifiers soft labels can refer to the class conditional probabilities, and a hard classification label can be produced based on the largest estimated probability.

$$y = \max_y\{P(Y = y|x), \;\; x \in X, \;\; y \in Y\} \tag{5}$$

For example, a sample tweet could be 65% likely to be happy, 18% likely to be relaxed, 9% likely to be angry, and 8% likely to be sad. Since the maximum probability of the tweet is 65%, it can be assigned to the happy class.

Naive Bayes and logistic regression are probabilistic classifiers which produce a probability distribution over output classes. Other models such as support vector machines do not produce probabilities. They instead return decision scores which are proportional to the distance from the separating hyperplane. They classify input data (here, tweets) with certain decision scores, which can be considered as soft labels. However, these scores may not correspond with class membership probabilities, since the distance from the separating hyperplane is not exactly proportional to the chances of class membership [**?**]. Some methods have been developed to convert the results of these classifiers into class membership probabilities. A common method is to apply Platt scaling [**?**], which learns the following sigmoid function defined by the parameters A and B on the decision scores $s(x)$:

$$P(Y = y|x) = \frac{1}{1 + e^{As(x)+B}} \tag{6}$$

Zadrozny and Elkan proposed another method by using isotonic regression when sufficient training data is available [**?**].

3.3 Emotex Experimental Results

In this experiment we collect enough labeled data to build emotion classifiers as described in Section 3.2.

3.3.1 Collecting Labeled Data and Building the Emotex Classifiers

To collect emotion-labeled data, we first identify a list of emotion hashtags as explained in Section 3.2.1. Using the list of keywords from the Circumplex model (see Figure 1), a set of

Class	Happy-Active	Happy-Inactive	Unhappy-Active	Unhappy-Inactive	Total
#Tweets before pre-processing	40000	41000	44000	41000	166000
#Tweets after pre-processing	34000	30000	37000	34000	135000

Table 2: Number of tweets collected as labeled data

emotion hashtags for each class was obtained. Then, we searched for the tweets containing these emotion hashtags and found more emotion hashtags from these tweets, such as the tag "#ifeelsad". At the end, a set of 20 unique emotion hashtags was collected for each emotion class. The objective was to assure that the tags of each class constitute emotions which are different compared with the emotions of the other classes. Using the identified hashtags, labeled data was collected for three weeks between December 26 and January 15. We used Twitter Stream API to collect data from online stream of tweets, which contains a 1% random sample of all tweets. Figure 5 presents the distribution of four classes of tweets that we labeled using hashtags during and after the new year vacation. It shows that the number of happy tweets after vacation is less than the number of happy tweets during vacation by about 13%. More interestingly, the number of unhappy tweets after vacation is more than twice the number of unhappy tweets during vacation. It also shows that the number of active tweets during the vacation are higher than the number of active tweets after vacation by about 4%.

To train our emotion classifiers we select equal size random samples for each emotion class from our collected labeled tweets. In fact, we do random under-sampling to create a balanced training dataset with equal number of samples in each class [**?**]. The number of samples in each emotion class is large enough to train classifiers. Table 2 represents the number of labeled tweets selected for each class before and after pre-processing. The removal of noisy tweets during pre-processing decreased the number of tweets by 19%. We explore the usage of different features (see Section 3.2.3). Table 3 lists the distribution of features in the collected data after pre-processing.

As described in Section 3.2.4.1 we utilize two types of classification including soft and hard classification. The emotion classification results using soft and hard classification are elaborated below.

3.3.2 Emotex: Hard Classification Results

We used two folds of our labeled data to train classifiers and one fold for testing. We used WEKA to train Naive Bayes, and decision tree models and we used SVM-light [55] with a

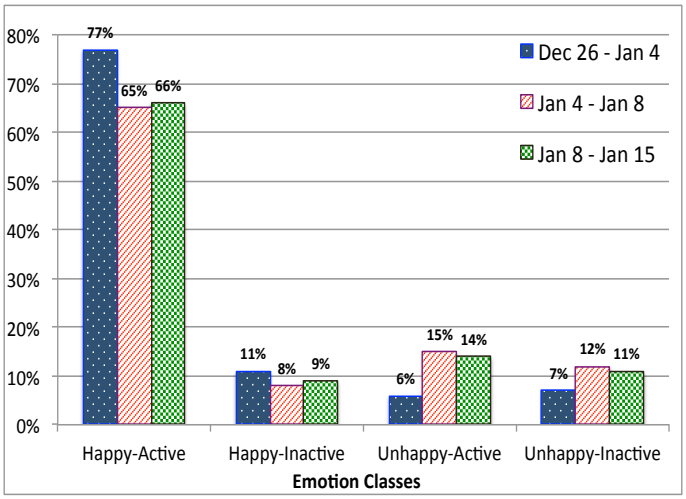

Figure 5: Distribution of four classes of emotion in collected tweets during and after the new year vacation.

Features	Happy icon	Sad icon	Angry icon	Sleepy icon	Negation	Punctuation
#Tweets with a feature	5800	1320	1020	270	9050	19450
%Tweets with a feature	4.3%	1%	0.7%	0.2%	6.7%	14.5%

Table 3: Distribution of features in the collected data

linear kernel to train the SVM classifier.

The classification results are evaluated in terms of F-measure ($\beta = 1$), defined as:

$$2 \times precision \times recall \; / \; (precision + recall)$$

Tables 4 presents precision, recall and F-measure of Naive Bayes, decision tree, and SVM using different features based on 3-fold cross validation. As it shows, decision tree achieved the highest accuracy using all the features. SVM achieved the highest accuracy using unigrams only, while Naive Bayes achieved the highest accuracy using unigrams and negations. Although a decision tree classifier provides high accuracy, it is slow. Therefore it is not practical for big datasets. SVM-light [55] runs fast and provides the highest accuracy.

The F-measure ($\beta = 1$) values of the SVM model in classifying four emotion classes using different features are presented in Figure 6. Class unhappy-active got the highest F-score. The active classes (i.e., happy-active and unhappy-active) achieved the highest F-score using unigrams only. However for the other classes the highest F-score is achieved using unigrams and punctuations. Across all emotion classes, the unigram-trained model

Features	Naive Bayes			SVM			Decision Tree		
	Prec.	Rec.	FM	Prec.	Rec.	FM	Prec.	Rec.	FM
Unigram	87.7	86.3	86.3	90.3	89.7	**90**	89.6	89.5	89.5
Unigram Emoticon	87.6	86.4	86.4	89.3	88.8	89	89.7	89.6	**89.6**
Unigram Punctuation	87.1	86.6	86.6	90.4	89.3	**89.9**	89.8	89.7	89.7
Unigram Negation	87.9	86.9	86.9	89.5	88.8	89.1	89.9	89.6	**89.7**
All-Features	87.3	87	86.9	90.2	89.5	**90**	90.1	89.9	**90**

Table 4: Precision, Recall and F-Measure ($\beta = 1$) of SVM, Naive Bayes and Decision Tree using different features

gave the highest overall F-score, and among other features punctuation performed second best.

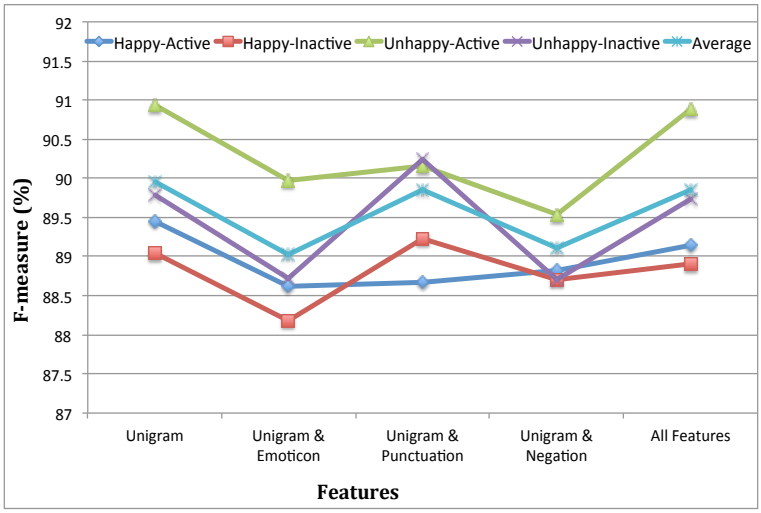

Figure 6: The F-measure ($\beta = 1$) of SVM model to classify four emotion classes using different features

3.3.3 Emotex: Soft Classification Results

We utilize a probabilistic classifier to measure the soft label based on the probability of assigning a tweet to each emotion class. In this experiment, we run Naive Bayes classifier on our training dataset and produce the class membership probabilities for each tweet.

Then the tweets whose maximum probability are higher than a predefined threshold are classified to the class with the maximum probability. The probability threshold is a tuning parameter of the system. We use the test set ROC curve to find a good probability threshold by resampling. For instance, the tweet "I can live for months on a good compliment." is 65% likely to belong to the happy-active class, 18% likely to belong to the happy-inactive class, 9% likely to belong to the unhappy-active class, and 8% likely to belong to the unhappy-inactive class. Since the maximum probability of this tweet is 65%, it therefore can be classified as a happy-active tweet. As another example, the tweet "Miss you already!" is 19% likely to belong to the happy-active class, 24% likely to belong to the happy-inactive class, 25% likely to belong to the unhappy-active class, and 32% likely to belong to the unhappy-inactive class. The maximum probability of this tweet is 32%, which is fairly small. Thus the tweet cannot be classified with a high enough certainty to render a hard classification.

Figure 7 shows the results of running Naive Bayes classifier on our labeled data with the probability threshold of 50% (Table 2 provides details about our labeled data). As it shows 81% of tweets are classified with the maximum probability higher than the threshold of 50%, where a hard label will be emerged by our system. Only 4% of these tweets are classified wrongly. However, 52% of the tweets whose maximum probability are lower than the threshold are classified inaccurately. In fact, tweets with low confident classification make an error rate of 52%, thus no hard label will be recommended by the system. The results confirm the fact that if tweets are classified with low certainty (i.e., low maximum probability), the classification results have a high error rate. This justifies our approach of forcing a hard classification only for a certain level of confidence.

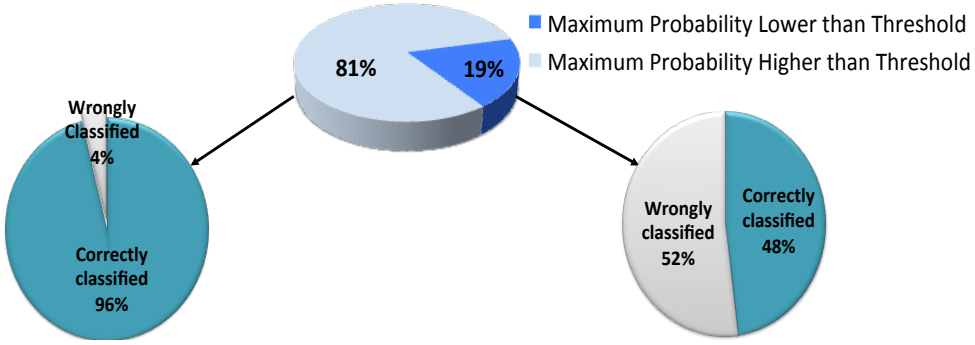

Figure 7: Distribution of classified tweets based on maximum probability with threshold =50%

Based on our observation shown in Figure 7 tweets with low maximum probability have a higher error rates, we thus separate them in our analysis. In fact, we only consider tweets that are classified with high maximum probability in our analysis.

	#Tweets	Precison	Recall	FMeasure
No Filtering	134,100	88%	83.7%	86%
Removing tweets with low max-probability	108,516	96%	94.4%	95.8%

Table 5: Classification results of Naive Bayes after removing tweets with low maximum probability

Table 5 shows the accuracy of classification before and after filtering out the tweets with maximum probability lower than the threshold of 50%. As it shows the accuracy has increased by 9.8%, after filtering out the tweets whose maximum probability scores are lower than the threshold of 50%.

3.3.4 Comparing Emotex with Lexical Approaches

Existing methods for text classification can be categorized into two main groups: lexical methods and supervised learning methods [66]. To further benchmark the performance of Emotex in classifying emotional messages, we compare it with the lexical approach.

The lexical approach has been previously studied in the context of emotion classification [112, 30, 32]. Lexical methods classify the emotion expressed in texts based on the occurrence of certain words. A lexicon of emotion words is created, in which each word belongs to an emotion class. Text messages are then classified using this emotion lexicon, typically by employing frequency counts of terms. The lexical methods may consider only terms of the lexicon directly or may associate numerical weights with these terms.

Lexical methods are based on shallow word-level analysis, and can recognize only surface features of the text. They usually ignore semantic features (e.g., negation) [66]. Moreover they rely on an emotion lexicon, which is difficult to construct a comprehensive set of emotion keywords. The creation of emotional lexicon is both time consuming and labor-intensive, and requires expert human annotators.

A variety of Emotion lexicons including ANEW lexicon [18], WordNet Affect [113], and LIWC dictionary [91] have been developed. To compare the results of Emotex with the lexical approach we utilize ANEW lexicon, which contains 2477 affect words that are rated for valence and arousal on a 1-9 scale. To classify messages using ANEW lexicon, the average valence and arousal of each message is estimated using the following formulas:

$$Valence_{tweet} = \frac{\sum_{i=1}^{n} v_i f_i}{\sum_{i=1}^{n} f_i} \tag{7}$$

$$Arousal_{tweet} = \frac{\sum_{i=1}^{n} a_i f_i}{\sum_{i=1}^{n} f_i} \tag{8}$$

Emotion Classes	Emotex			Lexical Method		
	Prec.	Rec.	FM	Prec.	Rec.	FM
Happy-Active	84.2	95.4	89.5	54.4	60.6	57.3
Happy-Inactive	94.3	84.4	89.1	64.2	58.5	61.2
Unhappy-Active	91.4	90.5	91	79.4	44	56.6
Unhappy-Inactive	91.2	88.4	89.9	91.5	52.5	66.7

Table 6: Comparing the classification results of Emotex with ANEW lexical approach based on precision, recall and F-measure ($\beta = 1$).

where n is the number of affect words occurring in the tweet, f_i is the frequency of the ith affect word, and v_i and a_i are the valence and arousal of the ith affect word respectively.

Then using the following heuristic the message can be easily classified: Less than 5 means low arousal/valence, more than 5 means high arousal/valence, and equal to 5 is neutral. For example, the tweet "Family and friends made this Christmas great for me." with the affect words family, friends and Christmas, the valence and arousal values are as following:

$$Valence = (7.74 + 7.65 + 7.8) \,/\, 3 \; = 7.73$$

$$Arousal = (5.74 + 4.8 + 6.27) \,/\, 3 \; = 5.60$$

Since both valence and arousal are larger than five, the tweet is labeled as happy-active. We compare the performance of Emotex with the lexical approach in classifying our labeled data shown in Table2. Table 6 lists the classification results of Emotex and the lexical approach in classifying different emotion classes. As the table shows, the F-score of Emotex is about 24% higher than the lexical approach utilizing ANEW lexicon.

3.4 Evaluating the Emotex Labeling Method

In the preceding sections, we have assumed that hashtags are true labels of the emotions expressed in text messages. However, the question still remains whether this assumption is correct. To answer this question, we need to determine whether human annotators would categorize texts into the same emotion classes selected by automatic labeling using hashtags. To evaluate the accuracy of hashtags as emotion labels, we performed two user studies in which two different classes of annotators participated. First, psychology experts (counselors and psychology graduate students) and then psychology novices (the crowd) were asked to classify texts into emotion classes.

3.4.1 Comparing Hashtag Labels with Crowdsourced Labels

This user study compares the accuracy of emotion labels that are created automatically using hashtags with labels made by non-expert annotators (the crowd). We design the study by randomly selecting 120 tweets (i.e., 30 tweets from each emotion class) from our collected emotion-labeled tweets (see Section 3.3.1). The tweets are shuffled to make their order random. Any embedded hashtags were removed from these 120 tweets as they are to serve as potential labels. Then the participants were asked to indicate the emotion expressed in each message by selecting the pleasure level (high for happy or low for unhappy), and the arousal level (high for active or low for inactive). We recruited labelers from the students in an introductory psychology class at Worcester Polytechnic Institute. Our user study was run online using the Qualtrics [1] survey system for three months. 60 students participated and 49 students completed the survey.

The perception of emotions expressed in texts tends to be subjective and diverse. As expected, inconsistencies occurred in the answers, such that in some cases different participants categorized the same text into different classes. Thus, we measure to what degree the participants agreed on the level of pleasure or activation of each tweet. We utilized Fleiss-Kappa to measure the level of agreement between a fixed number of labelers in classifying subjects. The Fleiss-Kappa value for inter-labeler agreement of the pleasure level of tweets was 0.67, which corresponds to a substantial agreement. This value for the activation level was 0.25 which shows a low level of agreement. In summary, although the annotators substantially agreed on the level of pleasure, there was a relatively low agreement among them for the level of activation. This conclusion can be explained by the fact that authors of text messages tend to express pleasure in explicit and unambiguous terms. For example, the tweet "Final weeks is going to be a death of me!" shows sadness. However it does't clearly indicate the level of arousal (i.e., activation).

The result of this study indicates that the labels created by non-experts to classify emotion in texts are not sufficiently reliable. This casts doubt on the use of the crowd (i.e., via Amazon Mechanical Turk) for this particular task of emotion classification. Note that participants in our study are a relatively notable crowd, as they are students in psychology that are trained to do user studies and have a general interest.

3.4.2 Comparing Hashtag Labels with Expert Labels

As the results of previous study indicates the level of agreement among crowd labelers is not sufficient to be able to consider them as ground truth especially for evaluating hashtag labels. Instead we sought the help of domain experts for labeling. We asked three psychol-

[1] http://www.qualtrics.com

Labeler	Pleasure level	Activation level
Crowd Labeler	0.67	0.25
Expert Labeler	0.84	0.64

Table 7: Comparing Fleiss-Kappa values of crowd and expert labelers

Expert	Counseling Director	Trained Expert1	Trained Expert2	Experts Consensus
Accuracy	81%	81%	84%	**88**%

Table 8: Accuracy of hashtag labels based on expert labels

ogy experts to manually label 120 tweets (the same set of tweets that had been utilized in Section 3.4.1). One of the experts is the director of counseling at WPI Student Development and Counseling Centre. The other two experts are graduate students in psychology who have been trained to classify emotions.

The Fleiss-Kappa measure of agreement between experts for the pleasure level of tweets is 0.84 which constitutes a high level of agreement. This value for the activation level is 0.64 which shows a substantial agreement. Table 7 lists the Fleiss-Kappa values of crowd labelers versus expert labelers. The agreement between experts is much higher than the agreement between crowd labelers. These results indicate that emotion labeling by trained experts is more reliable. It thus is more appropriate to be utilized as the ground truth. However, we note that if experts are used to label messages, crowdsourcing will be prohibitively expensive.

We now utilize the expert labels to evaluate the accuracy of hashtags. Table 8 lists the accuracy of hashtags based on expert labels. Hashtag labels are same as expert labels in 102 tweets. There are 14 tweets for which their hashtag labels are different from the expert labels. Also there is no consensus among experts about 4 tweets. Therefore, in about 88% of the cases, emotions indicated by hashtags embedded in tweets accurately captured the author's emotion indicated by the ground truth (i.e., expert labels). Most of the mismatches between hashtags and expert labels belong to the arousal level of tweets (i.e., active or inactive), which is not an intuitive concept to understand by non-psychologists.

3.5 Conclusion

In this research, we study the problem of automatic emotion detection in text messages. We develop and evaluate a supervised machine learning approach to automatically classify emotion in text messages. We develop a system called Emotex to create models for classifying emotion. Our experiments show that created models correctly classify emotion

in 90% of text messages. Created models were able to achieve about 90% accuracy for multi-class emotion classification.

We have investigated the use of hashtags in Twitter messages that indicate the emotion expressed by tweets as viable alternative to manual labeling. This approach overcomes the need for manual labeling and yields a completely automatic scheme for labeling a massive repository of Twitter messages. To address the problem of fuzzy boundary and variations in expression and perception of emotions, a dimensional emotion model is utilized to define emotion classes. Furthermore, a soft (fuzzy) classification approach is proposed to measure the probability of assigning a message into each emotion class.

4 DeepEmotex: A Deep Learning Approach to Detect Emotion in Text Messages

4.1 Introduction

Deep learning has emerged as a powerful machine learning technique that now has pushed the boundary of innovations in the field of Natural Language Processing and produced state-of-the-art results in many NLP applications. Applying deep learning to sentiment and emotion analysis has also become very popular recently.

For decades, machine learning methods solving NLP problems have been based on shallow models (e.g., SVM and logistic regression) trained on very high dimensional, sparse and hand-crafted features [133]. In the last few years, neural networks based on dense vector representations have been producing superior results on various NLP tasks shattering prior winning scores [133]. This trend is sparked by the success of word embeddings [78, 76] and deep learning methods [111]. Deep learning models automatically learn multiple layers of feature representations for NLP tasks and thus reduce the need for hand-crafted features [133].

Collobert et al. [5] demonstrated that a simple deep learning framework outperforms most state-of-the-art approaches in numerous NLP tasks including named-entity recognition (NER), semantic role labeling (SRL), and POS tagging. Since then, numerous complex deep learning based algorithms have been proposed to solve difficult NLP tasks.

Recent deep neural network-based approaches have achieved remarkable successes on text classification tasks by learning from hundreds of thousands of input data [6]. In order to obtain a model that performs well on a new dataset from another domain or task sufficient number of labeled examples is required by supervised learning methods. In some real-world applications, the labeled data may not be enough to learn an efficient classifier in the domain of interest. Transfer learning promises to improve this situation by transferring knowledge from a general-purpose source task to a more specialized target task. The main idea behind transfer learning is to borrow labeled data or extract knowledge from some related domains to help a machine learning algorithm to achieve greater performance in the downstream task [130].

The structure of this chapter is as follows: Section 4.2 reviews major deep learning related models and methods applied to natural language tasks including convolutional neural networks (CNNs) and recurrent neural networks (RNNs). Section 4.3 introduces the concept of distributed representation, the basis of sophisticated deep learning models. Section 4.4 describes the recent approaches of transfer learning, attention mechanisms and using transformer models for language-related tasks. Finally, Section 4.5 describes our

proposed approach, DeepEmotex, to detect emotion using transfer learning, following by the experimental and evaluation results in Section 4.6.

4.2 Background Knowledge about Neural Networks Models

Deep neural networks have achieved considerable success in varied tasks in text. Variations of Recurrent Neural Networks, such as Long Short Term Memory networks (LSTM) and Bidirectional LSTM (BiLSTM) have been effective in modeling sequential information [36]. Also, Convolutional Neural Networks (CNN) has proven their ability to decipher abstract concepts from raw signals in text domains [57].

4.2.1 Convolutional Neural Networks (CNNs)

CNN is a special type of feedforward neural network originally employed in the field of computer vision. Figure 8 shows a CNN model for an example sentence. In CNN, different filters are used to scan the input realized by convolution layers. Following the convolutional layer, a subsampling (or pooling) layer is usually used to reduce the dimension of the representation, this reduces the number of features and the computational complexity of the network [134]. After pooling layers, CNN uses a fully connected layer and then a softmax readout layer with output classes for classification.

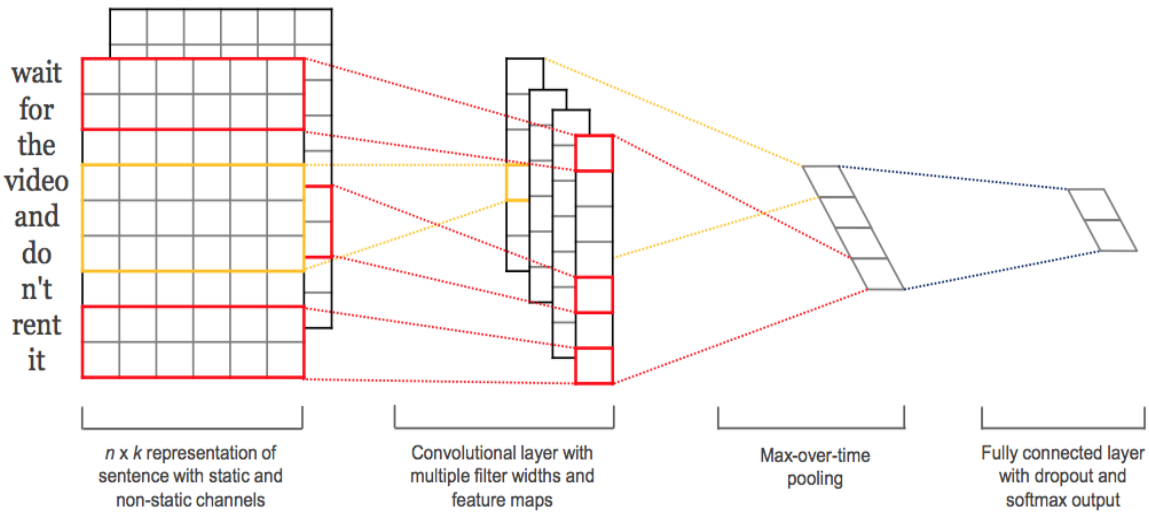

Figure 8: Convolutional neural network [57].

Convolutional layers in CNN play the role of feature extractor. The convolutional fil-

ters of a CNN model learn local features by enforcing a local connectivity between neurons of adjacent layers. As a result, CNN is useful for text classification as it finds strong local features regarding class membership. However, these features can appear in different places in the input. For example, in document classification a single key phrase (or an n-gram) can help in determining the topic of the document without knowing where they appear in the document. CNN can determine such discriminative phrases in a text with a max-pooling layer. Thus, convolutional and pooling layers help the CNN to extract these local indicators, regardless of their positions in sentences [134].

CNN models have achieved high performance in NLP tasks. CNN has the capability of capturing local correlations of spatial or temporal structures. For sentence modeling, CNN model achieves outstanding results in extracting n-gram features at different positions of a sentence through its convolutional filters [136].

4.2.2 Sequence Modeling: Recurrent Neural Networks (RNNs)

Recurrent neural networks or RNNs are a family of neural networks whose connections between neurons form a directed cycle. Therefore, they are a natural choice for sequence modeling tasks [118]. Figure 9 depicts the dynamics of the RNN network. It shows a simple RNN containing a single, self connected hidden layer. Unlike feedforward neural networks, RNNs can process a sequence of inputs, which makes it popular for processing sequential data [39]. An RNN can map from the previous inputs to each output. The key point is that the recurrent connections allow a memory of previous inputs to persist in the network's internal state, and therefore influence the network output. A useful method to visualise RNNs is to unfold the network along the input sequence. Figure 10 shows part of an unfolded RNN after one time step. Note that the unfolded graph (unlike Figure 9) contains no cycles. Each node represents a layer of network units at a single time step.

The information travels through the neural network from input neurons to the output neurons, while the error is calculated and propagated back through the network to update the weights. In RNNs the information travels through time, which means that information from previous time steps is used as input for the next time steps. The cost or the error function can be calculated at each time step. The gradient descent algorithm finds the global minimum of the cost function that is an optimal setup for the network. The problem is that the influence of a given input on the hidden layer, and therefore on the output, either decays or blows up exponentially as it cycles around the network's recurrent connections. This problem is often called the vanishing or exploding gradient problem [90]. It happens due to numeric underflow or overflow, i.e., when the multiplication of derivative terms during back-propagation become extremely small or very large.

In practice, standard RNNs suffer from the inability to learn long-term dependencies

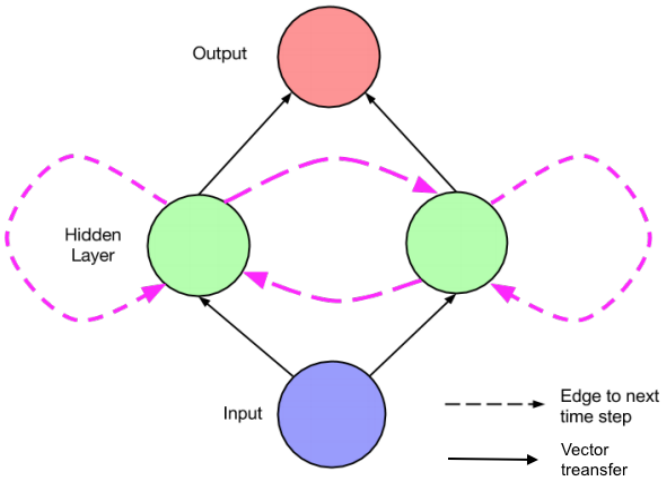

Figure 9: A simple recurrent net with one input unit, one output unit, and two recurrent hidden units [65].

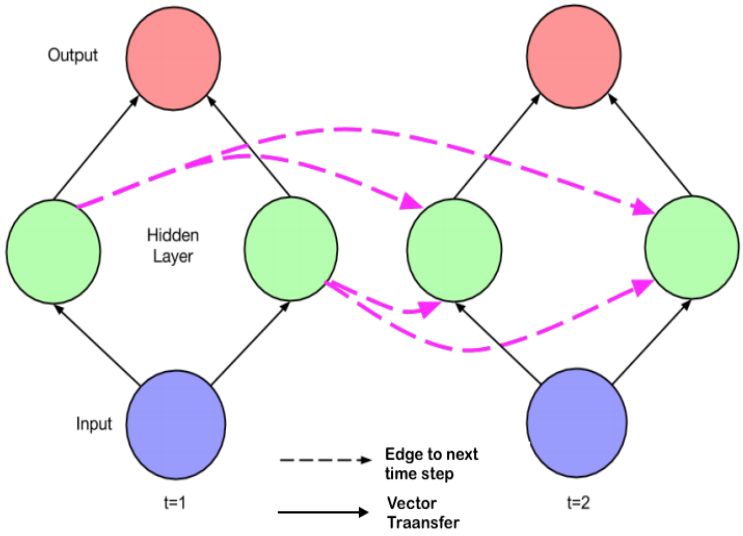

Figure 10: The recurrent network of Figure 9 unfolded across time with one time step [65].

due to vanishing/exploding gradient problem. In other words, when the length of the input sequence (i.e., the gap between two time steps) becomes large, RNNs are unable to remember the dependencies between inputs which are far apart in the sequence. To avoid the problem of vanishing (or exploding) gradient in the standard RNN, Long Short-term Memory RNN (LSTM) and other variants were designed for better remembering and memory accesses [136]. Different types of RNN have been developed to deal with the shortcomings of the standard RNN model including Bidirectional RNN, LSTM net-

work, and attention mechanisms. Their main purpose is to enable more memory size into RNNs. Specifically, attention mechanism accomplishes this by enabling direct connections between any two nodes far away, therefore reducing the length of flow.

4.2.3 Long Short-Term Memory (LSTM)

LSTM is a special type of RNN, which is capable of learning long-term dependencies. In order to cope with the problem of vanishing and exploding gradients, the LSTM architecture replaces the normal neurons in an RNN with LSTM cells with a little memory. LSTM is similar to a standard recurrent neural network with a hidden layer, except that the summation units in the hidden layer (see Figure 9) are replaced by memory units [65]. These cells are wired together and they have an internal state that helps to remember errors over many time steps.

Figure 11 illustrates an LSTM memory unit with a single cell. In the original LSTM cell, there are two gates: one learns to scale the incoming activation and one learns to scale the outgoing activation. The cell can thus learn when to incorporate or ignore new inputs and when to release the feature to other cells. The input to a cell is feeded into all gates using individuals weights [65].

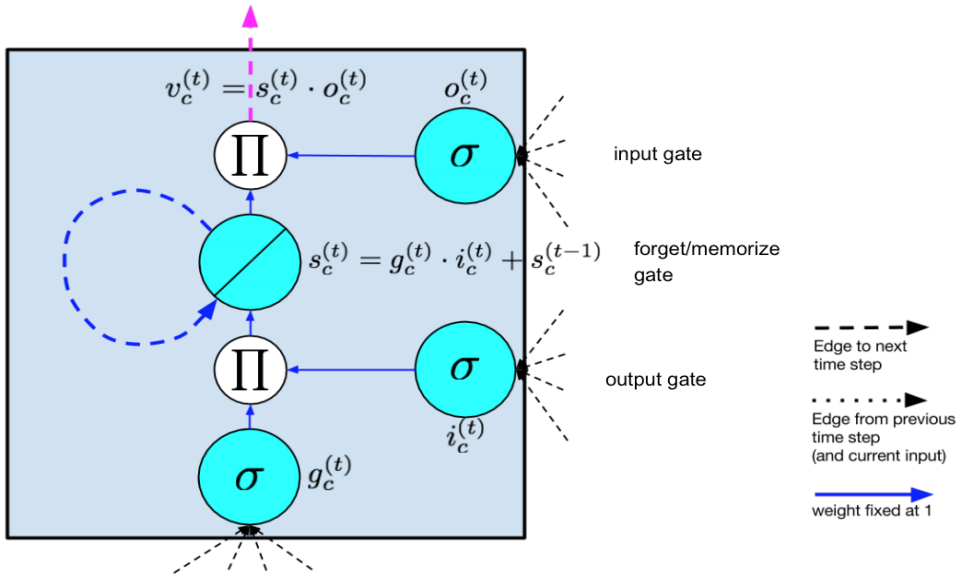

Figure 11: LSTM memory cell representation. The self-connected node is the internal state s. The diagonal line indicates that it is linear, i.e. the identity link function is applied. The blue dashed line is the recurrent edge, which has fixed unit weight. Nodes marked π output the product of their inputs. All edges into and from π nodes also have fixed unit weight [65].

LSTM was first introduced in 1997 by Hochreiter and Schmidhuber [50]. Over the

past decade, LSTM has re-emerged as a successful architecture for solving many sequence modeling tasks including many natural language processing tasks [39]. LSTM has proved successful at a range of various NLP tasks requiring long range memory, including learning context free languages [115], machine translation [116], and text classification [136].

4.3 Distributed Feature Representation

Text is a particular kind of data in which the word attributes are sparse and high dimensional. [3]. In order to train a classifier from labeled texts, words or sentences should be represented as a vector of numerical features. Feature representation plays an important role in text classification. Feature representation is needed to convert text content into a numeric vector representation, which can then be utilized to train classification models.

In this section, we will discuss some of the methods which are used for feature representation in text classification.

4.3.1 Word Representation

Distributed representation expresses a word as a dense real-valued vector of low dimension. Each word is positioned into a multi-dimensional space. In this model, words with similar meaning can be correspond to close vectors. Distributed representations (i.e., embeddings) are mainly learned through context. Distributed representation methods exploit word co-occurrences to build vectors of words. The main idea behind this approach is Distributional Hypothesis. According to the Distributional Hypothesis, words that appear in similar contexts tend to have similar meanings. Thus, distributed vectors try to capture the characteristics of the neighbors or context of a word. The main advantage of distributed vectors is that they capture similarity between words. Therefore, words with similar contexts have similar representations [71].

Distributed word representations (i.e., word embeddings) are dense and low-dimensional and real-valued vectors of words, with each factor in the vector representing some distinct informative feature of the word. Each entry in the vector represents a distinct informative property. These representations can capture semantic or syntactic regularities in the language [79].

Neural network models have been developed to learn low-dimensional vector representations of words. Recent works for learning vector representations of words using neural networks have succeeded in capturing fine-grained semantic and syntactic regularities [15, 26, 121, 75].

The vector representations are typically learned using neural networks as the underlying predictive model [15]. In this approach, each word is represented by a vector which is

concatenated or averaged with other word vectors in a context, and the resulting vector is used to predict other words in the context.

Distributed vectors or Word embeddings are often used as the first data processing layer in a deep learning model. Typically, these vectors are pre-trained by optimizing an auxiliary objective in a large unlabeled corpus, such as predicting a word based on its context [75, 78]. The word vectors learned using neural networks encode many linguistic and semantic patterns. Thus, these embeddings have proven to be efficient in capturing context similarity. Moreover, due to the smaller dimensionality of these these vectors, they are fast and efficient in computing core NLP tasks. Deep learning based NLP models represent words, phrases and even sentences using these embeddings.

Distributed representations developed in the context of statistical language modeling by Bengio et al. [15]. Authors proposed a neural language model which learned distributed representations for words. They argued that these word embeddings, once compiled into sentence representations using joint probability of word sequences, achieved an exponential number of semantically neighboring sentences.

Collobert et al. [26] developed a system using a convolutional architecture that shares representations across the tasks of language modeling, part-of-speech tagging, chunking, named entity recognition, semantic role labeling, and syntactic parsing. It was the first work to show the utility of pre-trained word embeddings as a useful tool for NLP tasks. The authors proposed a neural network architecture that forms the foundation to many current approaches. However, the immense popularization of word embeddings was due to [78]. Mikolov et al. [78] proposed the continuous bag-of-words (CBOW) and skip-gram models to efficiently construct high-quality distributed vector representations.

Another method for training word embedding is the Global Vectors model (GloVe), developed by Pennington et al. [92]. GloVe, builds a co-occurrence matrix for the entire corpus, which tabulates how frequently words co-occur with one another in a given corpus. GloVe is based on the word-context matrix [92], in which each row corresponds to a word and each column corresponds to a context. The element in the matrix is relevant to the co-occurrence times of the corresponding word and context. The source code for the GloVe model, as well as trained word vectors can be found at $nlp.stanford.edu/projects/glove/$.

Several natural language tasks such as reading comprehension and text classification have benefited from pre-trained word embeddings [78, 92] that are either fine-tuned for a specific task or held fixed. Below, we describe word2vec approach [78] to create word embeddings.

4.3.1.1 Word2vec

Word2vec is a popular implementation of neural network based algorithm for training the word embeddings developed by Mikolov et al. [78]. They introduced an efficient method for learning high-quality vector representations of words from large amounts of unstructured text data. Unlike most of the previously used neural network architectures for learning word vectors, training of word2vec embeddings does not involve dense matrix multiplications. This makes the training extremely efficient: an optimized single-machine implementation can train on more than 100 billion words in one day [75].

The word2vec vectors explicitly encode many linguistic and semantic patterns. Many of these patterns can be represented as linear translations, i.e., adding two word vectors results in a vector that is a semantic composite of the individual words. Figure 12 illustrates ability of the word2vec model to automatically learn the concepts and the relationships between them. During the training Mikolov et al. did not provide any supervised information about what a capital city means. For example, the result of a vector calculation vec("Madrid") - vec("Spain") + vec("France") is closer to vec("Paris") than to any other word vector. This compositionality suggests that a non-obvious degree of language understanding can be obtained by using basic mathematical operations on the word vector representations [78]. The code for word2vec is available at $code.google.com/p/word2vec/$ [75].

Word2vec trains a neural network to predict the n_{th} word given words $[1, ..., n_1]$ or the other way round. They proposed the continuous Bag-of-Words (CBOW) and Skip-Gram (SG) models to efficiently construct high-quality distributed vector representations. The CBOW model predicts a target word (e.g., "wearing") from its context words across a window of size k (e.g., "the boy is - a hat", where "-" denotes the target word). On the other hand, the skip-gram model does the inverse, by predicting the context words given the target word. The context words are assumed to be located symmetrically to the target words within a distance equal to the window size in both directions. In unsupervised settings, the word embedding dimension is determined by the accuracy of prediction. As the embedding dimension increases, the accuracy of prediction also increases until it converges at some point, which is considered the optimal embedding dimension as it is the shortest without compromising accuracy [133]. Statistically, the CBOW model smooths over a great deal of distributional information. It is effective for smaller datasets. However, the Skip-Gram model performs better for larger datasets [134].

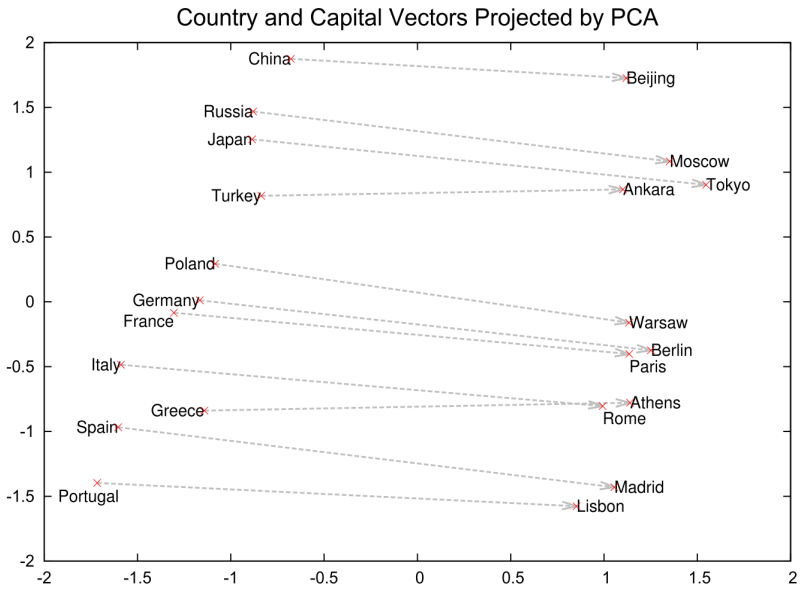

Figure 12: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities [78].

4.3.1.2 Continuous Bag of Words Model (CBOW)

The CBOW model is a simple fully connected neural network with one hidden layer (see Figure 13). The input layer consists of the one-hot encoded context words $x_1, x_2, ..., x_c$ in a word window of size C and vocabulary of size V. The hidden layer is an N-dimensional vector h. The output layer is output word y in the training example which is one-hot encoded. The one-hot encoded input vectors are connected to the hidden layer via a $V \times N$ weight matrix W and the hidden layer is connected to the output layer via a $V \times N$ weight matrix W'.

CBOW predicts a target word given its context. For this, the first step is to evaluate the output of the hidden layer. This is computed by:

$$h = \frac{1}{C} W (\sum_{t=1}^{C} x_i)$$

which is the average of the input vectors weighted by the matrix W. Next step is to compute the inputs to each node in the output layer:

$$u_j = {v'_{w_j}}^T \cdot h$$

where v'_j is the j^{th} column of the output matrix W'. Finally, the output of the output layer is computed. The output y_j is obtained by passing the input u_j throughout the soft-

max function:

$$y_j = P(w_{y_j}|w_1, w_2, ...w_c) = \frac{\exp(u_j)}{\sum_{j=1}^{V} \exp(u'_j)}$$

The Weight matrices W and W' are learned through back-propagation. First, the weight matrices are randomly initialized. Then, training examples are sequentially fed into the model and observe the error which is some function of the difference between the expected output and the actual output.

Thus, the first step in the process of learning the weight matrices W and W' is to define the loss function. The objective is to maximize the conditional probability of the output word given the input context, therefore the CBOW's loss function is as follows:

$$\mathcal{L}_{CBOW}(c, i) = -\log P(w_o|w_i)$$

$$\mathcal{L}_{CBOW}(c, i) = -u_j + \log \sum_{j=1}^{V} \exp(u'_j)$$

$$\mathcal{L}_{CBOW}(c, i) = -v_{w_o}^T \cdot h + \log \sum_{j=1}^{V} \exp({v'_{w_j}}^T \cdot h)$$

The next step is to compute gradient descent of this error with respect to the elements of both weight matrices and correct them in the direction of this gradient.

4.3.1.3 The Skip-gram Model

Skip-gram is used to predict the context words for a given target word in a sentence or a text [78]. Here, the target word is input while the context words are output. As there is more than one context word to be predicted the model becomes slightly complex. It can be simplified further by breaking down each (target, context_words) pair into (target, context) pairs such that each context consists of only one word.

In the skip-gram model (see Figure 14) x represents the one-hot encoded vector corresponding to the input word and $y_1, y_2, \ldots y_c$ are the one-hot encoded vectors corresponding to the output words in the training instance. The $V \times N$ matrix W is the weight matrix between the input layer and hidden layer whose i^{th} row represents the weights corresponding to the word in the vocabulary. This weight matrix contains the vector encodings of all of the words in the vocabulary (as its rows). Each output word vector also has an associated $V \times N$ output matrix W'. There is also a hidden layer consisting of N nodes. The output of the hidden layer h_i is simply the weighted sum of its inputs. Since the input

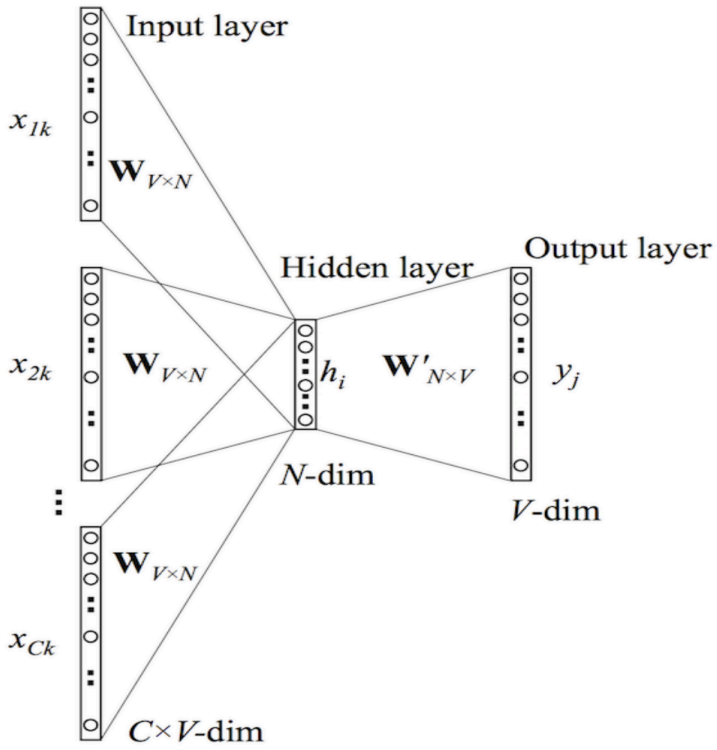

Figure 13: The CBOW model architecture: predicts the target word given the context words. [78].

vector x is one-hot encoded, the weights coming from the nonzero element will be the only ones contributing to the hidden layer. Therefore, the outputs of the hidden layer for the input x such that $x_k = 1$ and $x_{k'} = 0$ for all $k' \neq k$, will be equivalent to the k^{th} row of W.

$$h = x^T W = W_{(k,.)}$$

In the same way, the inputs to each of the output words is computed by the weighted sum of its inputs. Therefore, the input to the j^{th} node of each output word is:

$$u_j = {v'_{w_j}}^T \cdot h$$

where v'_j is the j^{th} column of the output matrix W'. Finally, the output of the output layer is computed. The output of the j^{th} node of the c^{th} output word can be computed via the following softmax function:

$$y_{c,j} = P(w_{c,j} = w_{o,c}|w_i) = \frac{\exp(u_{c,j})}{\sum_{j=1}^{V} \exp(u'_j)} \tag{9}$$

This value is the probability that the output of the j^{th} node of the c^{th} output word is

equal to the actual value of the j^{th} index of the c^{th} output vector.

So far, the inputs are propagated forward through the network to produce outputs. Now, the error gradients can be derived to learn both W and W' matrices via back-propagation. The first step in deriving the gradients is defining a loss function. This loss function will be:

$$\mathcal{L}_{Skip}(c, i) = -\log P(w_{o1}, w_{o2}, \dots w_{oc}|w_i)$$

where w_i is the input target word and $w_{o1}, w_{o2}, \dots w_{oc}$ are the context words within a window of size c. The loss function is simply the probability of the output words (the words in the input word's context) given the input target word. The objective of the skip-gram model is to maximize the log of this probability.

$$\mathcal{L}_{Skip}(c, i) = -\log \prod_{c=1}^{C} \frac{\exp(u_{c,j})}{\sum_{j=1}^{V} \exp(u'_j)}$$

$$\mathcal{L}_{Skip}(c, i) = -\sum_{c=1}^{C} u_{c,j} + C \log \sum_{j=1}^{V} \exp(u'_j)$$

With this loss function, the weight matrices W and W' are learned through back-propagation. First, the weight matrices are randomly initialized. Then, we compute the gradients with respect to the unknown parameters W and W' and at each iteration update them.

4.3.1.4 Negative Sampling

The softmax formulation is impractical because the cost of computing $P(w_{oj}|w_i)$ is proportional to the size of vocabulary $|V|$, which is often huge. Note that the summation over $|V|$ is computationally huge. This means summing across all words in the vocabulary is really expensive and will slow down training and testing of neural language models. Estimating the parameters of probabilistic models of language such as probabilistic neural models involves evaluating partition functions by summing over an entire vocabulary, which may be millions of words. Therefore estimating the parameters of such models become computationally expensive. [121].

Several approaches have been proposed to eliminate that linear dependency on vocabulary size and allow scaling to very large training corpora [121]. Noise Contrastive Estimation (NCE) by Mnih and Kavukcuoglu [80] and negative sampling by Mikolov et al. [75] are two related strategies in order to deal with this computational problem. Both papers propose scalable new approaches to learn word embeddings. Although they are superfi-

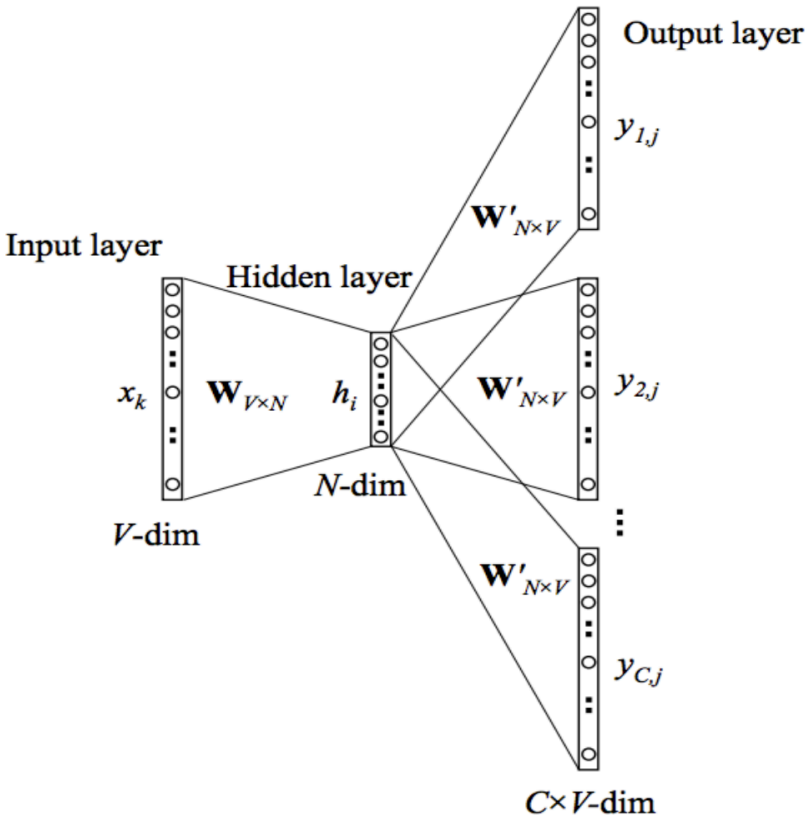

Figure 14: The Skip-gram model architecture: predicts the surrounding words given the current word. The training objective is to learn word vector representations that are good at predicting the nearby words [78].

cially similar, NCE is a general parameter estimation technique, while negative sampling is useful for learning word representations but not as a general-purpose estimator.

Noise Contrastive Estimation (NCE) is an alternative to the softmax, introduced by Gutmann and Hyvarinen [42], and applied to natural language modeling by Mnih and Teh [81]. NCE assumes that a good model should be able to differentiate data from noise using logistic regression.

While NCE can be shown to approximately maximize the log probability of the softmax, the skip-gram model is only concerned with learning high-quality vector representations, so negative sampling just simplifies NCE as long as the vector representations retain their quality. In fact, the objective of negative sampling is to learn high-quality word representations rather than achieving low perplexity on a test set, as is the goal in language modeling.

Mikolov et al. [78] proposed negative sampling to optimize the softmax formula and speed up training. For every training step, instead of looping over the entire vocabulary,

they just sample several negative examples. They approximate the ratio of Equation 9 using sigmoid functions and k randomly-sampled words, called negative samples. In this ratio, the numerator computes the similarity between the target word w and the context c (as measures by their dot product). The denominator computes the similarity between all other words in the vocabulary and the target word w. The objective is to maximize the ratio in the Equation 9 so that the words that appear closer together in text have more similar vectors than the words that do not. So, to make this computation cheap, instead of summing across all words they only sum across a few negative words i.e words that don't belong to the same context as the target word. They sample from a noise distribution whose probabilities match the ordering of the frequency of the vocabulary.

The paper [78] says that selecting 5-20 words (i.e., negative samples) works well for smaller datasets, and only 2-5 words (i.e., negative samples) should be enough for large datasets. An example for negative sampling is that if *volleyball* appears in the context of *sports*, then the vector of *sports* is more similar to the vector of *volleyball*, than the vectors of several randomly chosen words (e.g. math, democracy, queen, animal), instead of all other words in the vocabulary.

Let's consider a pair (w, c) of a word and a context. Did this pair come from the training data? Let's denote the probability that (w, c) come from the corpus data by $p(D = 1|w, c)$. Correspondingly, $p(D = 0|w, c) = 1 - p(D = 1|w, c)$ is the probability that (w, c) is not in the corpus data (i.e., it is noise) [38]. Let's assume there are parameters θ controlling the distribution: $p(D = 1|w, c; \theta)$. The goal is now to find parameters to maximize the probability that a word and a context being in the training data if it indeed is, and maximize the probability that a word and a context not being in the training data if it indeed is not. We take a simple maximum likelihood approach of these two probabilities. By assuming the probability that the target and context words co-occur to be independent of their position in the training data, this objective can be computed as:

$$arg\max_{\theta} \prod_{(w,c)\in D} p(D = 1|w, c; \theta) \prod_{(w,c)\in D'} p(D = 0|w, c; \theta)$$

$$= arg\max_{\theta} \log \prod_{(w,c)\in D} p(D = 1|w, c; \theta) \prod_{(w,c)\in D'} (1 - p(D = 1|w, c; \theta))$$

$$= arg\max_{\theta} \sum_{(w,c)\in D} \log p(D = 1|w, c; \theta) + \sum_{(w,c)\in D'} \log(1 - p(D = 1|w, c; \theta))$$

The quantity $p(D = 1|c, w; \theta)$ can be defined using sigmoid function:

$$p(D = 1|w, c; \theta) = \frac{1}{1 + e^{-v_c.v_w}}$$

The resulting objective is:

$$= arg\max_{\theta} \sum_{(w,c)\in D} \log \frac{1}{1+e^{-v_c.v_w}} + \sum_{(w,c)\in D'} \log(1 - \frac{1}{1+e^{-v_c.v_w}})$$

$$= arg\max_{\theta} \sum_{(w,c)\in D} \log \frac{1}{1+e^{-v_c.v_w}} + \sum_{(w,c)\in D'} \log \frac{1}{1+e^{v_c.v_w}}$$

If we let $\sigma(v_c.v_w) = \frac{1}{1+e^{-v_c.v_w}}$ we get:

$$= arg\max_{\theta} \sum_{(w,c)\in D} \log \sigma(v_c.v_w) + \sum_{(w,c)\in D'} \log \sigma(-v_c.v_w) \tag{10}$$

Note that D' is a "negative" corpus (i.e., a corpus with false sentences such as "stock boil fish is toy").

As you see it converted to a binary classification task ($y = 1$ as positive class, $y = 0$ as negative class). As we need labels to perform this binary classification task, we designate all context words c (i.e., all words in window of the target word) as true labels ($y = 1$, positive sample), and k randomly selected from corpora as false ($y = 0$, negative sample).

The Equation 10 presents the negative sampling objective of Mikolov et al. for the entire corpus $D \cup D^{'}$. Their equation presents it for one example $(w, c) \in D$ and k examples $(w, c_j) \in D^{'}$ (see Equation 4 in Mikolov et al. paper [78]). Thus the task is to distinguish the target word w from draws from the noise distribution using logistic regression, where there are k negative samples for each data sample. The experiments indicate that values of k in the range $5 - 20$ are useful for small training datasets, while for large datasets the k can be as small as $2 - 5$.

Both NCE and negative sampling have the noise distribution as a free parameter. Mikolov et al. investigated a number of choices for the noise distribution and found that the unigram distribution $U(w)$ raised to the power of $3/4$ (i.e., $U(w)^{3/4}/Z$) outperformed significantly the unigram and the uniform distributions. Therefore, in order to selsct k negative samples, Mikolov et al. construct k samples $(w, c_1), ..., (w, c_k)$ for each $(w, c) \in D$, where each c_j is drawn according to its unigram distribution raised to the $3/4$ power, as shown below:

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=1}^{W} f(w_j)^{3/4}}$$

The main difference between the Negative sampling and NCE is that NCE needs both samples and the numerical probabilities of the noise distribution, while Negative sampling uses only samples. While NCE approximately maximizes the log probability of the

softmax, this property is not important for learning word representations. [78].

4.3.2 Sentence Representation

Similar to word embeddings (i.e., Word2Vec, GloVE, ELMo, or Fasttext), sentence embeddings embed a full sentence into a vector space. These sentence embeddings inherit features from their underlying word embeddings [9]. In practice, a sentence embedding might look like the following example:

"Rest in piece Kobe" = [0.2 , 0.1 , -0.3 , 0.9 , ...]

Researchers have developed different methods to compute embeddings that capture the semantics of word sequences (i.e., phrases, sentences, and paragraphs). They developed methods ranging from simple additional composition or averaging of the word vectors to sophisticated architectures such as convolutional neural networks and recurrent neural networks [9].

The current trend to embed word sequences or sentences is toward development of neural network architectures including and recursive networks based on non-syntactic hierarchical structure [135]. Other neural network structures such as convolution neural networks have been studied to create sentence embeddings [57, 53, 47]. Recurrent neural networks (RNNs) using long short-term memory (LSTM) capture long-distance dependency and have also been used for modeling sentences [118].

Most work in supervised learning for NLP builds task-specific representations of sentences rather than general-purpose ones [114]. Many neural NLP systems are initialized with pre-trained word embeddings but learn their representations of words in context from scratch, in a task-specific manner from supervised learning [57, 53]. For example, some researchers used pre-trained word embeddings for sentiment analysis [110, 119, 95]. However, learning these representations reliably from scratch is not always feasible, especially in low-resource settings. Some recent work has addressed this by learning general-purpose sentence representations [58, 127, 49, 28, 74, 86, 22, 31].

Wieting et al. [127] learned general-purpose, paraphrastic sentence embeddings by starting with standard word embeddings and modifying them based on supervision from the Paraphrase pairs dataset (PPDB), and constructing sentence embeddings by training a simple word averaging model. This simple method leads to better performance on textual similarity tasks than a wide variety of methods and serves as a good initialization for textual classification tasks. However, supervision from the paraphrase dataset seems crucial, since they report that simple average of the initial word embeddings does not work very well.

Encoder-decoder models have been successful at learning semantic representations [69]. Skip-Thought [58] trains an RNN encoder-decoder architecture to predict the sur-

rounding sentences of an encoded passage. Skip-Thought model consists of an encoder RNN that produces a vector representation of the source sentence and a decoder RNN that sequentially predicts the words of adjacent sentences. skip-thought vectors are an extension of the skip-gram model for word embeddings [78] to sentences. Skip-thought vectors learn re-usable sentence representations from weakly labeled data to capture similarity in terms of discourse context. Unfortunately, these models take weeks or often months to train [114, 69]. Arora et al. [9], however, demonstrate that simple word embedding averages are comparable to more complicated models like skip-thoughts. Hill et al. [49] showed, that the task on which sentence embeddings are trained significantly impacts their quality.

InferSent [28] uses labeled data of the Stanford Natural Language Inference (SNLI) dataset [17] and the MultiGenre NLI dataset [128] to train a BiLSTM network with max-pooling over the output. Conneau et al. [28] showed that InferSent consistently outperforms unsupervised methods like SkipThought.

Encoder-decoder based sequence models are known to work well, but they are slow to train on large amounts of data. On the other hand, bag-of-words models train efficiently by ignoring word order [114, 69]. Another work for learning sentence representations proposed by Logeswaran and Lee [69] by training an encoder architecture faster than previous methods. Given an input sentence, their model chooses the correct target sentence from a set of candidate sentences, instead of generating the target sentence. It has used objectives to rank candidate next sentences, given a representation of the previous sentence.

Universal Sentence Encoder [22] trains a transformer network and augments unsupervised learning with training on SNLI dataset. It showed a significant improvement by leveraging the Transformer architecture, which is based on attention mechanisms. Previous works [28, 22] found that SNLI dataset is suitable for training sentence embeddings.

BERT (Bidirectional Encoder Representations from Transformers) [31] is a bi-directional pre-training model back-boned by the Transformer network, a deep hybrid neural network with feed forward layers and self-attention layers. In prior work [69], only sentence embeddings are transferred to down-stream tasks, however BERT transfers all parameters to initialize end-task model parameters. BERT achieved state-of-the-art results for various NLP tasks, including sentence classification, Semantic textual Similarity, and sentence-pair regression.

Recently, Reimers and Gurevych developed t Sentence-BERT (SBERT) [103] to create sentence embeddings. They fine-tuned SBERT on MultiGenre NLI dataset [128]. SBERT embeddings significantly outperform other state-of-the-art sentence embedding methods like InferSent [28] and Universal Sentence Encoder [22].

4.4 Transfer Learning

Existing supervised algorithms work well only if the training and test data are represented by the same features and drawn from the same distribution. Furthermore, the performance of these algorithms heavily rely on collecting high quality and sufficient labeled training data to train a statistical model to make predictions on the unlabeled data. When the distribution changes, most statistical models need to be rebuilt using newly collected labeled data. However, it is expensive or impossible to collect the needed training data and rebuild the models in many real world applications. This problem has become a major bottleneck of making machine learning methods more applicable in practice [88].

In the real world, we observe many examples of transfer learning. For example, learning to play the electronic organ may help learning the piano. The study of transfer learning is motivated by the fact that people can intelligently apply knowledge learned previously to solve new problems faster and more efficient. In machine learning applications, it is expensive or impossible to collect sufficient training data to train models to use in each domain of interest. It would be more practical if we could reuse the training data and knowledge that is already extracted from some related domains/tasks to learn a precise model to use in the domain of interest. In such cases, knowledge transfer or transfer learning between tasks or domains become more crucial and desirable to reduce the need and effort to re-collect the training data.

Transfer learning can be largely beneficial in text classification. It would be helpful if we could transfer the text classification knowledge into the a new domain. One application where the need for transfer learning may arise is sentiment classification. Let's consider the task of classifying the reviews on a product, into positive and negative reviews. For this classification task, we need to first collect many reviews of the product and annotate them. Since the distribution of review data among different types of products can be different, to maintain good classification performance, we need to collect a large amount of labeled data for each product in order to train the review classification models. However, collecting enough labeled data for various products can be expensive. To reduce the effort for annotating reviews for various products, we may adapt a classification model that is trained on some products to learn classification models for some other products. In such cases, transfer learning can save a significant amount of labeling effort.

4.4.1 Transfer Learning Overview

The effectiveness of transfer learning has given rise to a diversity of approaches. As shown in Figure 15, transfer learning aims to borrow labeled data or extract the knowledge from one or more related source tasks and applies the knowledge to solve a target task [88].

Transfer learning, allows the domains, tasks, and distributions used in training and testing to be different. It can be defined as below:

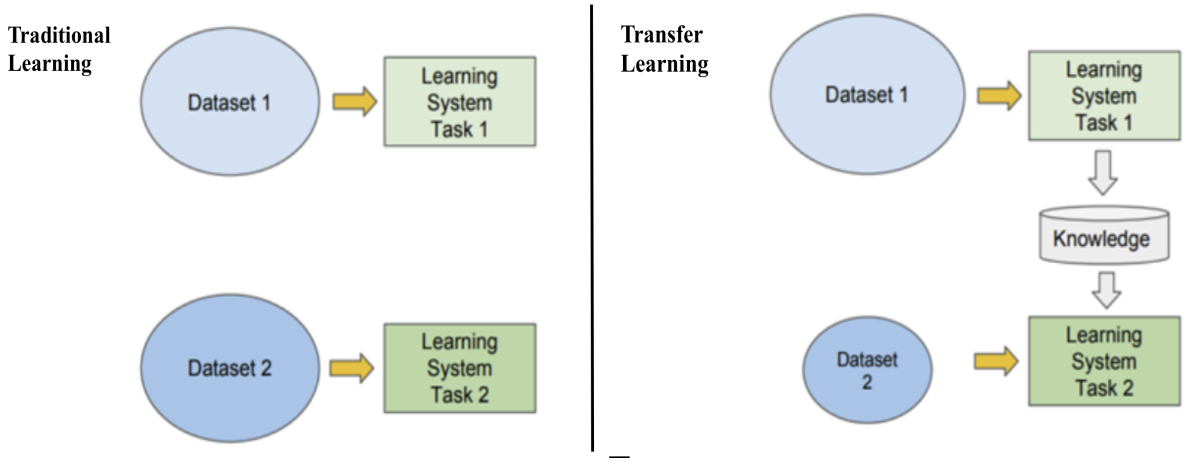

Figure 15: Traditional learning Versus Transfer Learning [88].

Definition 1. *(Transfer Learning): Given a source domain D_S and learning task T_S, a target domain D_T and a learning task T_T, transfer learning aims to improve learning of the target predictive function $f_T(\cdot)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$.*

In the above definition, a domain is a pair $D = \{X, P(x)\}$. Thus, $D_S \neq D_T$ implies that either (1) the feature spaces between the source dataset and the target dataset are different, i.e. $X_S \neq X_T$, or (2) their distributions are different, i.e. $P(x_S) \neq P(x_T)$. As an example, in text classification, case (1) corresponds to when the two sets of documents are described in different languages, and case (2) may correspond to when the source domain documents and the target domain documents focus on different topics [88, 70, 8].

Similarly, a task is defined as a pair $T = \{Y, P(y|x)\}$. Thus, the condition $T_S \neq T_T$ implies that either (1) the label spaces between the domains are different, i.e. $Y_S \neq Y_T$, or (2) the conditional probability distributions between the domains are different; i.e. $P(Y_S|X_S) \neq P(Y_T|X_T)$. In text classification example, case (1) corresponds to the situation where source domain has binary document classes, whereas the target domain has multi classes to classify the documents to. Case (2) corresponds to the situation where the source and target documents are very unbalanced in terms of the user-defined classes [88].

When the target and source domains are the same, i.e. $D_S = D_T$, and their learning tasks are the same, i.e., $T_S = T_T$, the learning problem becomes a traditional machine learning problem. Based on whether the feature spaces or label spaces are identical or not, we can further categorize transfer learning into two settings: 1) homogenous transfer learning, and 2) heterogenous transfer learning [130].

Homogeneous and heterogenous transfer learning can be defined as follows:

Definition 2. *(Homogeneous Transfer Learning): Given a source domain D_S and learning task T_S, a target domain D_T and a learning task T_T, homogeneous transfer learning aims to improve learning of the target predictive function $f_T(\cdot)$ in D_T using the knowledge in D_S and T_S, where $X_S \cap X_T \neq \varnothing$, and $Y_S = Y_T$, but $P(x_S) \neq P(x_T)$ or $P(y_S|x_S) \neq P(y_T|x_T)$.*

Based on the above definition, the feature spaces between domains are overlapping, and the label spaces between tasks are identical in homogeneous transfer learning. The difference between domains or tasks is caused by the marginal distributions or predictive distributions. Approaches to homogeneous transfer learning can be summarized into three categories: 1) instance transfer approach, 2) feature-representation transfer approach, 3) model-parameter transfer approach, and 4) relational-knowledge transfer approach [88, 130].

Recently, some researchers have already started to consider transfer learning across heterogeneous feature spaces or non-identical label spaces. heterogeneous transfer learning can be defined as follows[130]:

Definition 3. *(Heterogeneous Transfer Learning): Given a source domain D_S and learning task T_S, a target domain D_T and a learning task T_T, heterogeneous transfer learning aims to improve learning of the target predictive function $f_T(\cdot)$ in D_T using the knowledge in D_S and T_S, where $X_S \cap X_T = \varnothing$, or $Y_S \neq Y_T$.*

Based on the definition, heterogeneous transfer learning can be further categorized into two contexts: 1) transferring knowledge across heterogeneous feature spaces, and 2) transferring knowledge across different label spaces [130].

4.4.2 Transfer Learning Methods

The two most common transfer learning techniques in NLP are feature-based transfer and fine-tuning. Features-based transfer involves pre-training real-valued embeddings vectors. These embeddings may be at the word level [78] or sentence level [22]). The embeddings are then fed to custom downstream models.

Fine-tuning approaches involve copying the weights from a pre-trained network and tuning them on the downstream task. Recent work shows that fine-tuning often enjoys better performance than feature-based transfer [52].

Both feature-based transfer and fine-tuning require a new set of weights for each task. Fine-tuning is more parameter efficient if the lower layers of a network are shared between tasks. Consider a function (neural network) with parameters w as $\phi_w(x)$. Feature-based transfer composes ϕ_w with a new function, X_v, to yield $X_v(\phi_w(x))$. Only the new, task

specific, parameters, v, are then trained. Fine-tuning involves adjusting the original parameters, w, for each new task, limiting compactness.

4.4.2.1 Instance Transfer Approach

The first approach is transferring Knowledge of Instances, which assumes that certain parts of the data in the source domain can be reused for learning in the target domain by re-weighting. Instance re-weighting and importance sampling are two major techniques in this context [88, 130].

A motivation of the instance-transfer approach is that although the source domain labeled data cannot be reused directly, part of them can be reused for the target domain after re-weighting or re-sampling. An assumption behind this approach is that the source and target domains have a lot of overlapping features, which means that the domains share similar support. Based on whether labeled data are required or not in the target domain, the instance transfer approach can be further categorized into two contexts: 1) no target labeled data are available, and 2) a few target labeled data are available [130, 88].

In the first context of instance-transfer approach, no labeled data are required but a lot of unlabeled data are assumed to be available in the target domain. In this context, most instance-transfer methods are deployed based on an assumption that $P_S(y|x) = P_T(y|x)$, and motivated by importance sampling [130, 88].

In the second context of the instance-transfer approach, a few target labeled data are assumed to be available. Different from the first context, in this context, most approaches are proposed to weight the source domain data based on their contributions to the classification accuracy for the target domain [130, 88].

4.4.2.2 Feature Representation Transfer Approach

A second case can be referred to as feature-representation transfer approach. As described in the previous section, for the instance transfer approach, a common assumption is that the source and target domains have a lot of overlapping features. However, in many real-world applications, the source and target domains may only have some overlapping features, which means that many features may only have support in either the source or target domain. In this case, most instance transfer methods may not work well. The feature-representation transfer approach is promising to address this issue.

An intuitive idea behind the feature-representation transfer approach is to learn a good feature representation for the source and target domains such that based on the new representation, source domain labeled data can be reused for the target domain. In this sense, the knowledge to be transferred across domains is encoded into the learned feature rep-

resentation. With the new feature representation, the performance of the target task is expected to improve significantly. Specifically, the feature-representation-based approach aims to learn a mapping $\phi(\cdot)$ such that the difference between the source and target domain data after transformation, $\phi(x_{S_i})$'s and $\phi(x_{T_i})$'s, can be reduced. In general, there are two ways to learn such a mapping $\phi(\cdot)$ for transfer learning. One is to encode specific domain or application knowledge into learning the mapping, the other is to propose a general method to learn the mapping without taking any domain or application knowledge into consideration [130, 88].

4.4.2.3 Parameter Transfer Approach

The first two categories of approaches to transfer learning are in the data level. The instance-transfer approach tries to reuse the source domain data after re-sampling or re-weighting, while the feature-representation-transfer approach aims to find a good feature representation for both the source and target domains such that based on the new feature representation source domain data can be reused. Different from these two categories of approaches, a third category of approaches to transfer learning can be referred to as the model-parameter-transfer approach, which assumes that the source and target tasks share some parameters. A motivation of this approach is that a well-trained source model has captured a lot of structure, which can be transferred to learn a more precise target model. In this way, the transferred knowledge is encoded into the model parameters [130, 88].

4.4.3 Transformer

Early results on transfer learning for NLP leveraged recurrent neural networks [52, 94], but it has recently become more common to use models based on the Transformer architecture [122]. The standard Transformer architecture proposed by Vaswani et al. [122]. The Transformer was initially shown to be effective for machine translation, but it has subsequently been used in a wide variety of NLP settings due to its improved performance. Transformer models attain state-of-the-art performance in many NLP tasks, including translation, question answering, and text classification problems [122, 31, 99].

The primary building block of the Transformer is self-attention. Self-attention is a variant of attention [12] that processes a sequence by replacing each element by a weighted average of the rest of the sequence. Figure 16 shows the model architecture of the original transformer [122]. The encoder-decoder Transformer implementation originally-proposed by Vaswani et al. [122]. The original Transformer consisted of an encoder-decoder architecture and was intended for sequence-to-sequence tasks [116].

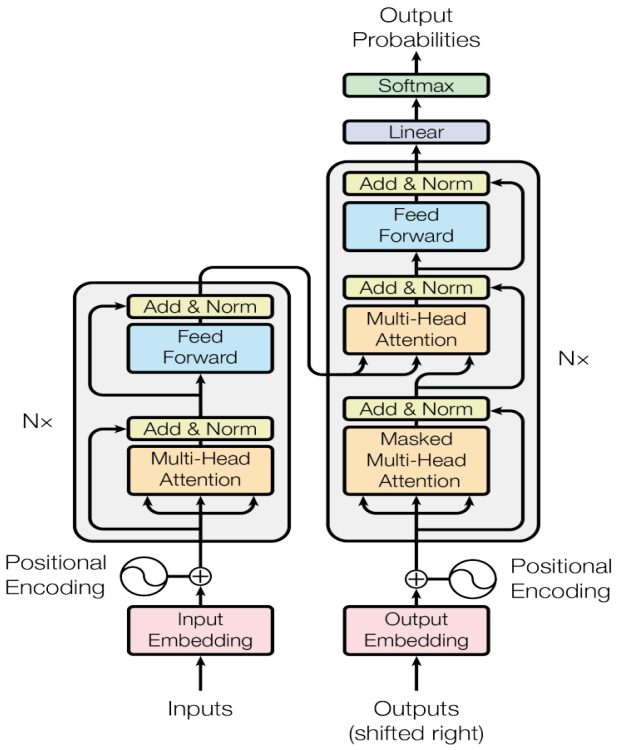

Figure 16: Model architecture of a Transformer [122].

Figure 17 shows a Transformer of 2 stacked encoders and decoders [2]. First, an input sequence of tokens is mapped to a sequence of embeddings, which is then passed into the encoder. The encoder consists of a stack of encoders. Each encoder is broken down into two sub-components: self-attention and feed-forward neural network. The encoder's inputs first flow through a self-attention layer. Self-attention layer helps the encoder look at other relevant words in the input sentence as it encodes a specific word. The outputs of the self-attention layer are fed to a feed-forward neural network. The exact same feed-forward network is independently applied to each position. Layer normalization [10] is applied to the input of each sub-component. Then the outputs are send out upwards to the next encoder [102].

The decoder is similar in structure to the encoder except that it includes a standard encoder-decoder attention mechanism after each self-attention layer that attends to the output of the encoder. The encoder-decoder attention layer helps the decoder focus on relevant parts of the input sentence. The self-attention mechanism in the decoder also uses a form of auto-regressive or causal self-attention, which only allows the model to attend to past outputs. The output of the final decoder block is fed into a dense layer with a

[2] http://jalammar.github.io/illustrated-transformer/

softmax output, whose weights are shared with the input embedding matrix. All attention mechanisms in the Transformer are split up into independent "heads" whose outputs are concatenated before being further processed [102].

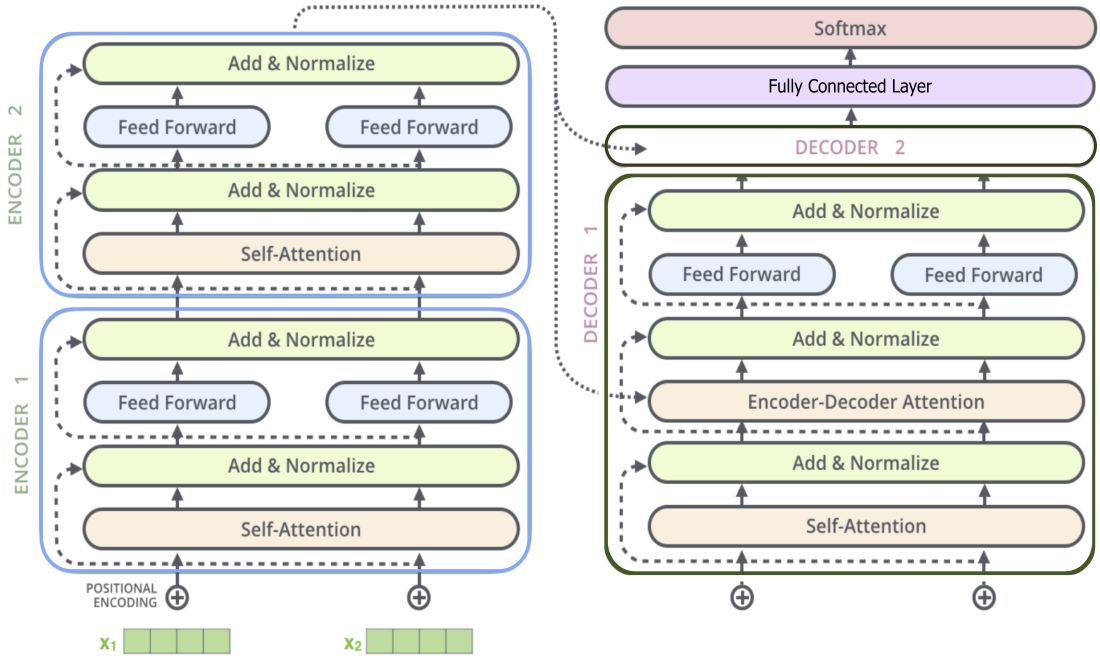

Figure 17: A Transformer of 2 stacked encoders and decoders.

While the Transformer was originally introduced with an encoder-decoder architecture, recent works on transfer learning for NLP use alternative architectures. It has recently also become common to use models consisting of a single Transformer layer stack, with varying forms of self-attention used to produce architectures appropriate for language modeling [99] or classification and span prediction tasks [31, 131].

Below, we describe two methods that utilize Transformers for text classification including BERT [31] and Universal Sentence Encoder [22].

4.4.3.1 Universal Sentence Encoder (USE)

Google recently introduced Universal Sentence Encoder [22]. The Universal Sentence Encoder encodes text into high dimensional vectors. The input is variable length English text and the output is a 512-dimensional vector. The model is trained and optimized for sentences, phrases or short paragraphs. Universal Sentence Encoder uses a transformer-network that is trained on a variety of data sources and a variety of tasks with the aim of dynamically accommodating a wide variety of natural language understanding tasks. A

pre-trained version has been made available for TensorFlow.

The Universal Sentence Encoder [22] is more powerful than word embedding, and it is able to embed not only words but phrases and sentences. As illustrated in Figure 18, the sentence embeddings can be used to compute sentence level semantic similarity scores that achieve excellent performance on the semantic textual similarity (STS) Benchmark [22]. When included within larger models, the sentence encoding models can be fine tuned for specific tasks using gradient based updates [22].

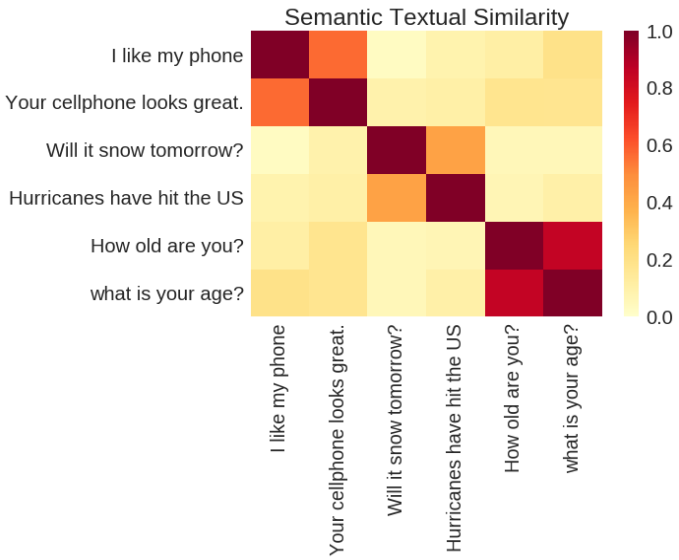

Figure 18: Sentence similarity scores using embeddings from the Universal Sentence Encoder [22].

As illustrated in Figure 19, there are two Universal Sentence Encoders with different encoder architectures to achieve distinct design goals. The first encoder is based on the transformer architecture [122], which aims for high accuracy at the cost of greater model complexity and more resource consumption. The other encoder targets efficient inference with slightly reduced accuracy by using a deep averaging network (DAN) [54]. In this model the embeddings for words and bi-grams are averaged together and then used as input to a deep neural network that computes the sentence embeddings.

The Transformer model constitutes an encoder and decoder. The encoder is composed of a stack of N = 6 identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. They also employed a residual connection around each of the two sub-layers, followed by layer normalization [122]. The transformer based encoder achieves the best overall transfer task performance. However, this comes at the cost of computing time and memory usage scaling dramatically with sentence length.

Deep Averaging Network(DAN) is much simpler where input embeddings for words and bi-grams are first averaged together and then passed through a feedforward deep neural network (DNN) to produce sentence embeddings. The primary advantage of the DAN encoder is that compute time is linear in the length of the input sequence.

Both models were trained with the Stanford Natural Language Inference (SNLI) corpus. The SNLI corpus is a collection of 570k human-written English sentence pairs manually labeled for balanced classification with the labels entailment, contradiction, and neutral, supporting the task of natural language inference (NLI). The sentence embeddings can be used to compute sentence-level semantic similarity scores. Essentially, the models were trained to learn the semantic similarity between the sentence pairs.

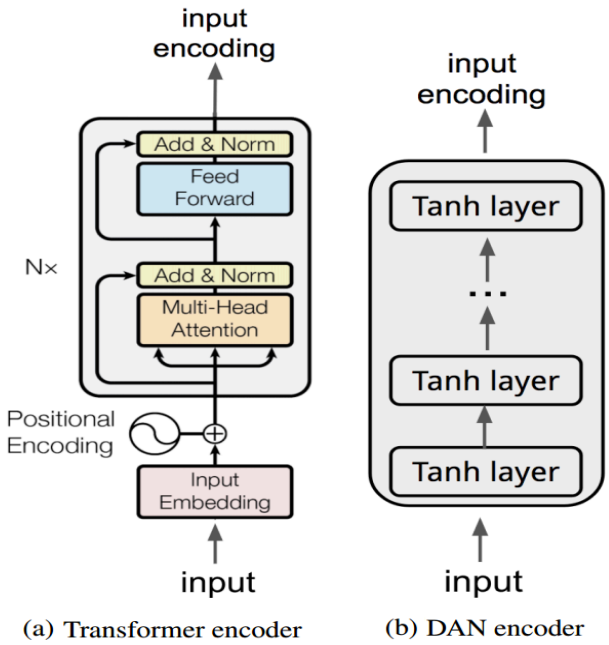

Figure 19: Model architectures comparison for the Transformer and DAN sentence encoders [22].

4.4.3.2 Bidirectional Encoder Representations from Transformers (BERT)

BERT [31] is another pre-trained model that has achieved state-of-the-art performance in many NLP tasks. BERT is one of the key innovations in the recent progress of contextualized representation learning [31, 52, 94, 99]. The idea behind the progress is that even though the word embedding [92, 78] layer (in a typical neural network for NLP) is trained from large-scale corpora, training a wide variety of neural architectures that encode contextual representations only from the limited supervised data on end tasks is insufficient.

Unlike ELMo [94] and ULMFiT [52] that are intended to provide additional features for a particular architecture that bears humans understanding of the end task, BERT adopts a fine-tuning approach that requires almost no specific architecture for each end task. This is desired as an intelligent agent should minimize the use of prior human knowledge in the model design. Instead, it should learn such knowledge from data [129].

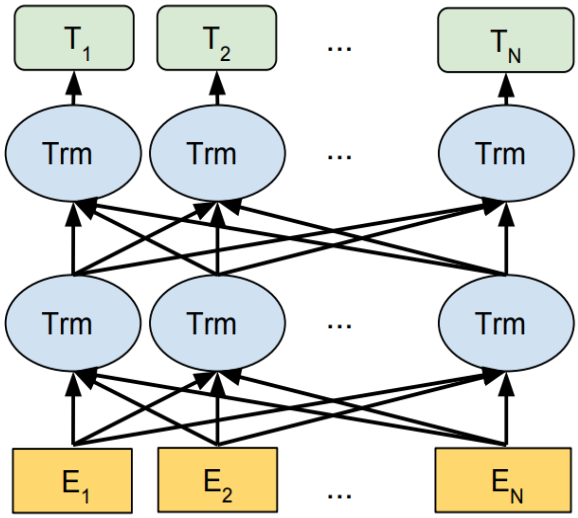

Figure 20: Pre-training model architecture of BERT [31]. E_i is the embedding representation, T_i is the final output and Trm is the intermediate representations of the same token.

As shown in Figure 20 BERT's model is a multi-layer bidirectional Transformer encoder based on the original implementation proposed by Vaswani et al. [122]. A word starts with its embedding representation from the embedding layer. Every layer does some multi-headed attention computation on the word representation of the previous layer to create a new intermediate representation. All these intermediate representations are of the same size. In a 12-layers BERT model a token will have 12 intermediate representations.

BERT adapts pre-training to capture dependencies between sentences via a next sentence prediction task as well as by constructing training examples of sentence-pairs with input markers that distinguish between tokens of the two sentences [11].

The output embedding of BERT captures the general linguistic information from the large and general dataset it was trained on. A model can also be fine-tuned to generate an embedding which is more contextually aware of a new domain. The output of this form of fine tuning would be another embedding. The goal of fine tuning is thus an embedding that contains domain specific information.

BERT has two parameter intensive settings: BERTBASE and BERTLARGE. BERTBASE includes 12 layers, 768 hidden dimensions and 12 attention heads (in transformer) with

the total number of parameters, 110M. BERTLARGE contains 24 layers, 1024 hidden dimensions and 16 attention heads (in transformer) with the total number of parameters, 340M.

BERT considers only Books Corpus and Wikipedia data sources to pre-train their models, [11]. Although BERT aims to learn contextualized representations across a wide range of NLP tasks (to be task-agnostic), leveraging BERT alone still leaves the domain challenges unresolved (as BERT is trained only on formal texts and has almost no understanding of social media text) [129].

BERT [31] and RoBERTa [68] has set a new state-of-the-art performance on sentence-pair regression tasks like semantic textual similarity. However, it requires that both sentences are fed into the network, which causes a massive computational overhead: Finding the most similar pair in a collection of 10,000 sentences requires about 50 million inference computations ($\simeq 65$ hours) with BERT [103]. The construction of BERT makes it unsuitable for semantic similarity search as well as for unsupervised tasks like clustering [103].

4.5 DeepEmotex: A Deep Learning Approach to Classify Emotion in Text using Sequential Transfer Learning

Detecting emotions in text is a challenging problem because of the semantic ambiguity of the emotion expression in text and fuzzy boundaries of emotion classes [45]. Moreover, the context can completely change the emotion for a sentence as compared to perceived emotion when the sentence is evaluated standalone. For example, the sentence "I started crying when I realized!" will be perceived as a sad feeling, however considering it in the context "I just qualified for the scholarship. I started crying when I realized!", it turns out to be a happy emotion. Similarly, the sentence "Try to do that again!" is very likely to be perceived as no-emotion, however a majority will judge it as an Angry feeling with the context "How dare you make fun of me like that! Try to do that again!". Therefore, considering context is important to detect emotion of a text.

Feature selection and representation are important tasks for emotion classification in texts due to the high dimensionality of text features and the existence of irrelevant features. Performance of classification algorithms highly depends on feature selection and representation [14]. Traditional machine learning systems are based on hand-crafted features on NLP tasks. However, creating a complete set of hand-crafted features is very tedious and time-consuming. Moreover, using a comprehensive set of hand-crafted features lead to a huge feature space with sparse and high dimensional feature vectors and is not efficient.

In contrast, deep learning models automatically learn multiple layers of low dimen-

sional feature representations and thus reduce the need for hand-crafted features. Deep learning methods have been utilized in learning word representations through neural language models [78]. Along with the success of deep learning in many other application domains, deep learning is also popularly used in sentiment and emotion classification in recent years [133].

4.5.1 DeepEmotex: A Sequential Transfer Learning Model

As described in Section 3, Emotex system [45] exploits the usage of static keywords with explicit emotional values as emotion-specific features. These features are selected from predefined sets of emotion lexicons including ANEW lexicon [18] and LIWC dictionary [91] as described in Section 3.2.3. However, the list of emotion keywords may change in different domains. Thus, Emotex requires extensive hand-crafted features to achieve high performance due to diverse ways of representing emotions in different domains. Such hand-crafted features are time-consuming to create and they are often incomplete. To solve this problem, we develop a deep learning approach called DeepEmotex to be able to extract dynamic features based on context instead of using static hand-crafted features.

While deep learning models have achieved state-of-the-art results on many NLP tasks, these models are trained from scratch, requiring huge datasets, and days to converge [52]. Instead of learning from scratch, transfer learning can be used to transfer knowledge from a general-purpose domain and task into a more specialized target domain and task [102, 51, 106]. Transfer learning matches the performance of training a deep learning model from scratch, with much less labeled examples.

Transfer learning can be beneficial in emotion classification. Supervised learning methods have proven to be promising in emotion classification. However, these methods are domain dependent, which means that a model built on one domain (e.g., messages on a specific topic or event) may perform poorly on another domain. The reason is that different domain-specific words may be used to express emotion in different domains. Table 9 shows common keywords of two different domains including, death of George Floyd and Covid-19 epidemic. For example, keywords *justice, protest, violent, murder, racism* are domain-specific words of the first domain whereas keywords such as *death, disease, fever* are Covid19-specific keywords. Due to the mismatch of common keywords between different domains, an emotion classifier trained on one domain may not work well when directly applied to other domains. Therefore, cross-domain emotion classification algorithms are highly desirable to reduce domain dependency and manually labeling costs by transferring knowledge from related domains to the domain of interest [88].

DeepEmotex utilizes sequential transfer learning approach, where source and target tasks are learned in sequence. That means, models are not optimized jointly as in multi-

Death of George Floyd	attack, violence, protest, curfew, death, murder, racism, black lives
Covid-19 Pandemic	fever, death, patient, disease, vaccine, mask, immunity, epidemic

Table 9: Example keywords of different domains

task learning but each task is learned separately [106]. Sequential transfer learning consists of two stages: a pre-training phase in which general-purpose representations are learned on a source task or domain, and an adaptation or fine-tuning phase during which the learned knowledge is transferred to a target task or domain [88].

Most prior work has focused on different pre-training objectives to learn general-purpose word or sentence representations [78, 58]. A few works have explored the fine-tuning phase and how to adapt the pre-trained model to a given target task. There are two common approaches for fine-tuning: The first approach is to use the pre-training network as a feature extractor [33], where all layers in the model are frozen when fine-tuning on the target task except the last layer. In this approach the pre-trained representations are used in a downstream model. Alternatively, the pre-trained model's parameters are unfrozen and fine-tuned on a new task [29]. This approach enables to adapt a general-purpose representation to many different tasks.

Figure 21 shows our DeepEmotex model. Our model represents input words by their embeddings. Following the embedding layer, our model consists of a transformer encoder, followed by a SoftMax classification layer. Gaining a better understanding of the adaptation phase is key in making the most use out of pre-trained representations. Accordingly, DeepEmotex utilizes two state-of-the-art pre-trained models, known as BERT [31] and Universal Sentence Encoder [22]. Using these models, we transfer knowledge learned from a large corpus to our emotion classification model. We then fine-tune DeepEmotex model to fit to our target emotion datasets.

4.5.2 DeepEmotex: A Transfer Learning Model using Universal Sentence Encoder

Universal Sentence Encoder (USE) [22] is a deep neural network to create universal sentence embeddings. Universal embeddings are pre-trained embeddings obtained from training deep learning models on a huge corpus. These pre-trained (generic) embeddings can be used in a wide variety of NLP tasks including text classification, semantic similarity and clustering.

The Universal Sentence Encoder is trained and optimized for greater-than-word length text, such as sentences, phrases or short paragraphs. The input is a variable length text. The Universal Sentence Encoder encodes the input text into 512-dimensional embeddings. The

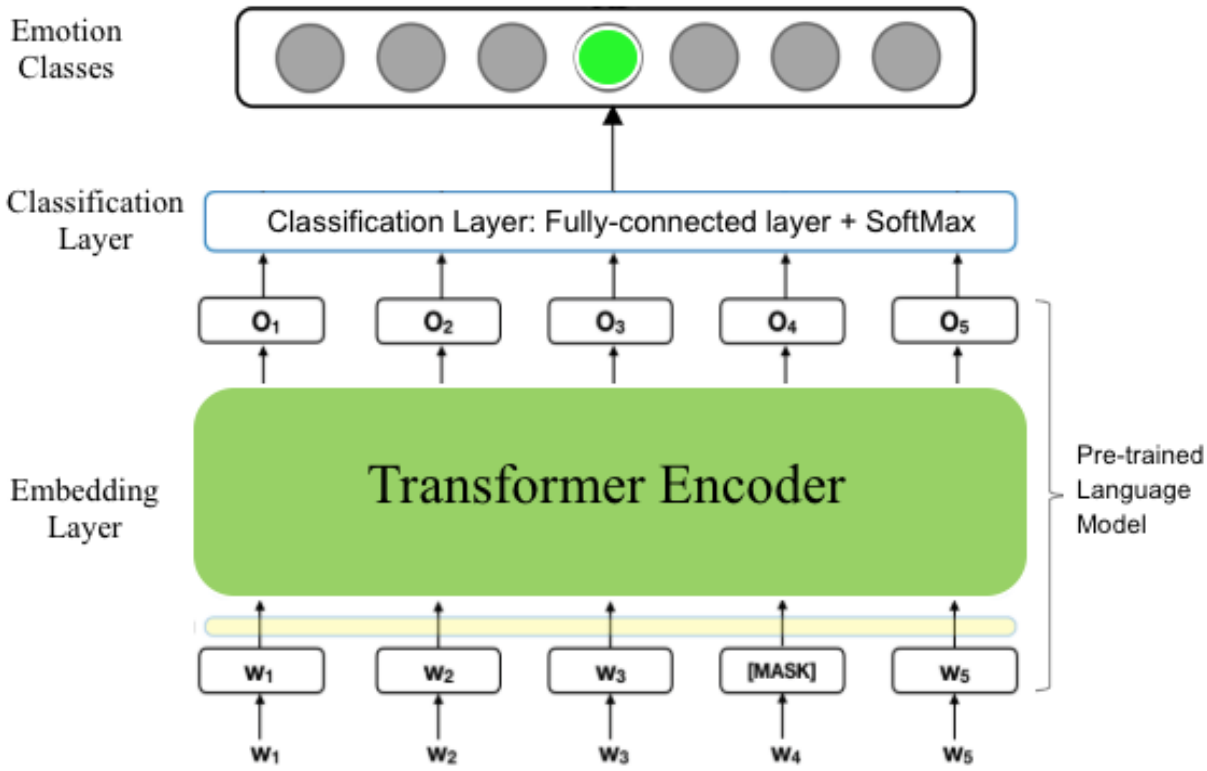

Figure 21: Model of DeepEmotex. The embedding layer learns an embedding that contains the semantic textual information in input text. The learned representations are fed into the classification layer for emotion prediction.

embeddings are trained on different data sources and tasks with the aim of dynamically accommodating a wide variety of natural language understanding tasks which require modeling the meaning of word sequences rather than just individual words.

The USE sentence encoding models are trained on a variety of unsupervised training data including Wikipedia, web news, web question-answer pages and discussion forums. The USE models augment unsupervised learning with training on supervised data from the Stanford Natural Language Inference (SNLI) corpus. The SNLI corpus is a collection of 570k human-written English sentence pairs manually labeled for classification with the labels entailment, contradiction, and neutral, supporting the task of natural language inference.

Their key finding [22] is that transfer learning using sentence embeddings tends to outperform word embedding level transfer [22]. Essentially, there are two versions of the USE models. The first version makes use of a Deep Averaging Network (DAN) where input

embeddings for words and bi-grams are first averaged together and then passed through a feed-forward deep neural network (DNN) to produce sentence embeddings [22]. Deep Averaging Network (DAN) is simpler than the second version. The primary advantage of the DAN encoder is that its compute time is linear in the length of the input sequence.

The second version makes use of the transformer-network based sentence encoding model. The transformer encoder is composed of a stack of N = 6 identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network, followed by layer normalization.

Their results [22] demonstrate that the transformer-based encoder achieves the best overall transfer task performance. However, this comes at the cost of computing time and memory usage scaling dramatically with sentence length. For our emotion classification task we use the transformer-based encoder as it achieves better overall performance than the DAN encoder.

We first define the sentence embedding sub-network that leverages the Universal Sentence Encoder before we describe the design of our emotion classification model. We apply transfer learning leveraging prior knowledge from pre-trained embeddings to solve a new task. We utilize transfer learning to fine-tune the sentence embeddings using our collected emotion-labeled dataset.

We build a feed-forward neural network with two hidden dense layers and the rectified linear activation function (ReLU). ReLU is a piece-wise linear function with a constant derivative. ReLU overcomes the vanishing gradient problem. This is good for deep neural networks which suffer from the vanishing and explosion gradient problem. ReLU is easy to compute, fast to converge in training and yields good performance in neural networks [134].

A dense layer is a fully connected layer, meaning all the neurons in a layer are connected to those in the next layer. A dense layer provides learning features from all the combinations of the features of the previous layer. The input of our model is 512-feature vectors created using the Universal Sentence Encoder technology. The resulting vector is then fed into fully connected layers culminating in a softmax layer. We then fine-tune our model using collected labeled tweets introduced in Section 4.6.1. We fine-tune the embedding weights by setting the trainable parameter to true. Here we leverage transfer learning in the form of pre-trained embeddings.

The overall model architecture is shown in Figure 22. The input of the model is a twitter message. First, the embedding layer uses the pre-trained USE model to map a sentence into its embedding vector. The model that we are using splits the sentence into tokens, embeds each token and then combines them into context-aware 512-dimension

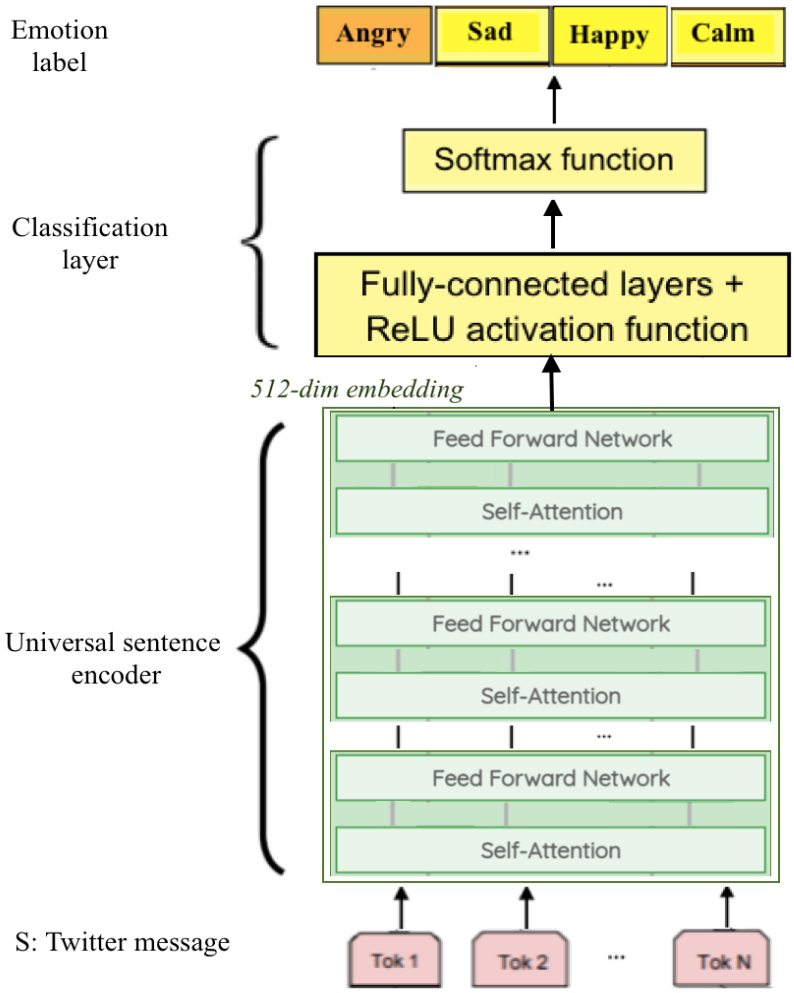

Figure 22: Model of DeepEmotex to classify emotion in the text messages using Universal Sentence Encoder.

embeddings. Then, the embeddings are passed through a feed-forward neural network with ReLU activation. It projects the input into 256-dimension embeddings and feeds them to the classification layer to produce a classification probability. The output of our model is an emotion classification label. The main objective is to correctly predict the emotion of each tweet.

4.5.2.1 DeepEmotex: Experimental Setup of the Model using Universal Sentence Encoder

Universal Sentence Encoder model can be fine-tuned to our target task in several ways by freezing layers to disable parameters updates. One common approach is to use the network as a feature extractor [33], where all layers in the model are frozen during fine-tuning on the target task except the last layer. Another common approach is to use the pre-trained model as an initialization, and thereafter the full model is unfrozen [36]. We implement both approaches to fine-tune Universal Sentence Encoder. Our first transfer learning approach is implemented using the following workflow:

- Instantiate a base model and load pre-trained weights into it.
- Freeze all layers in the base model by setting trainable = False.
- Add classification layer (Fully connected layer + SoftMax) on top of the frozen layers.
- Train the new layers on our new dataset.

Next, we implement the second approach. For this, we unfreeze all layers and re-train the whole model on our dataset for several epochs. This helps fine-tune the model towards our task by incrementally adapting the pre-trained features to our new data. We fine-tune the embedding weights by setting trainable=true. Weights are updated (via gradient descent) to minimize the loss during training. By training all layers of the model we are able to adjust the parameters across the network during back propagation. Our second transfer learning approach is implemented using the following workflow:

- Instantiate a base model and load pre-trained weights into it.
- Unfreeze all layers in the base model by setting trainable = True.
- Add classification layer (Fully connected layer + SoftMax) on top of the USE unfrozen layers.
- Train all the layers on our new dataset to fine-tune the old parameters on the new dataset.

We use the Universal Sentence Encoder Version 3 [3] as our base model. We add a feed-forward neural network with two hidden layers and the Relu activation function.

We set our batch size to 150 and number of epochs to 20. Batch-size defines the number of examples will be passed to our model during one iteration, and number of epochs is the

[3]https://tfhub.dev/google/universal-sentence-encoder-large/3

number of times our model will go through the entire training set. We train the model on our training datasets using Adam optimizer with a learning rate of 0.001. The performance of the re-trained model is evaluated at the end of training epochs with our test datasets.

4.5.3 DeepEmotex: A Transfer Learning Model using Bidirectional Encoder Representations from Transformers

The model architecture of DeepEmotex using Bidirectional Encoder Representations from Transformers (BERT) is shown in Figure 24. The input of the model is a twitter message and the output is an emotion label. We use the pre-trained BERT model [31] to generate text representations. BERT learns text representations using a bidirectional Transformer encoder [122] pre-trained on the language modeling task. Transformers have a sequence-to-sequence model architecture. The model of each transformer is shown in Figure 16. Each transformer includes a separate encoder and decoder component. The difference is in their use of attention known as self-attention. The core architecture consists of a stack of encoders fully connected to a stack of decoders. Each encoder consists of a self-attention component and a feed forward network. Each decoder consists of a self-attention component, an encoder-decoder attention component, and a feed forward component.

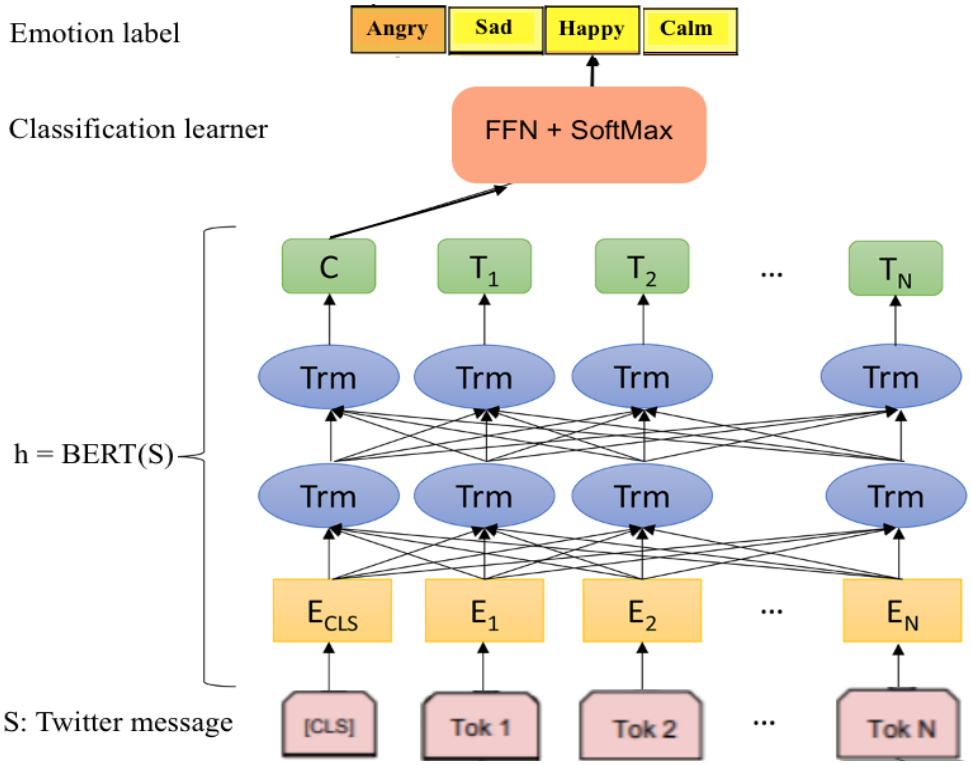

Figure 23: Model of DeepEmotex to classify emotion in text messages using pre-trained BERT.

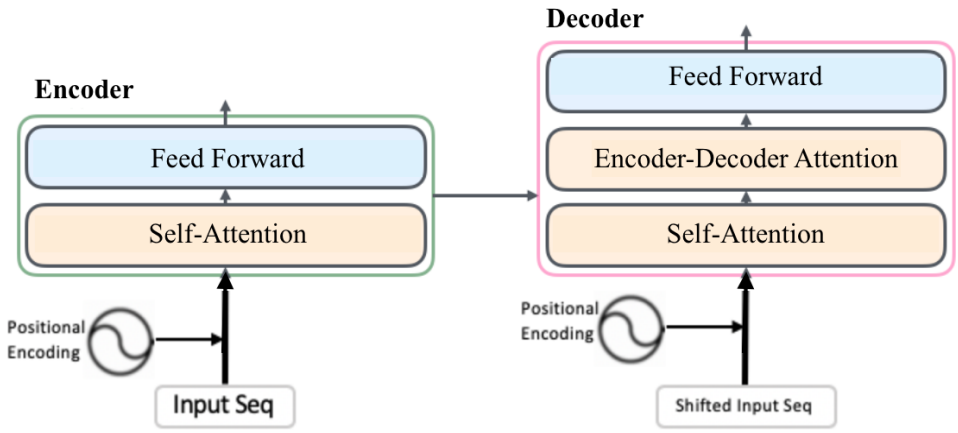

Figure 24: Model of Transformer.

4.5.3.1 DeepEmotex: BERT as the Encoding Layer

BERT has several variants based on model configurations. We adopt BERT-base [122] as our base model. BERT-base consists of an encoder with 12-layer Transformer blocks. For each block in the encoder, it contains a 12-head self-attention layer and 768-dimensional hidden layer, yielding a total of 110M parameters. The base model allows inputs up to a sequence of 512 tokens and outputs the vector representations of the sequence. The input sequence has one or two segments. The first token of the sequence is always $[CLS]$ which contains the special classification embedding. Another special token $[SEP]$ is used for separating segments. In order to facilitate the training and fine-tuning of BERT model, we transform the input text into $[CLS] + text + [SEP]$ format.

4.5.3.2 DeepEmotex: Target-specific Classification Layer

We follow Devlin et al. [31] and create a fully-connected layer over the final hidden state corresponding to the $[CLS]$ input token. During fine-tuning, we optimize the entire model, with the additional softmax classifier parameters $W \in R^{K \times H}$, where H is the dimension of the hidden state vectors and $K = 4$ is the number of emotion classes.

Let $S = ([CLS], t_1, \ldots, t_m, [SEP], ..., t_n)$ be the input sequence (i.e., Twitter message), where $t_1, \ldots, t_m$ denotes a sentence with m tokens. For emotion classification task, we use the final hidden state $h = BERT(S)$ of the first token $[CLS]$ as the representation of the whole sequence S. A standard softmax classifier added on top of BERT predicts the probability of emotion label c:

$$x = W \cdot h + b$$

$$P(c|x) = softmax(x) = \frac{\exp(x)}{\sum_{k=1}^{C} \exp(x)}$$

where W is the task-specific weight matrix, and b is the bias vector to be estimated. We fine-tune all parameters as well as W jointly by maximizing the log-probability of the correct label.

4.5.3.3 DeepEmotex: Fine-tuning BERT

BERT uses books corpus and Wikipedia data sources to pre-train their models [11]. Although BERT aims to learn contextualized representations across a wide range of NLP tasks, leveraging BERT alone still leaves the domain challenges unresolved as BERT is only trained on formal texts and has almost no understanding of social media text [129]. The end tasks from the original BERT paper typically use tens of thousands of examples to ensure that the system is task-aware [129]. Below, we introduce fine-tuning BERT to boost the performance of classifying emotion on Twitter messages.

We adopt the pre-trained BERT-BASE and ROBERTA-BASE as the encoding layer of our DeepEmotex model. We extend them with extra tasks-specific layers and fine-tune the model on our emotion classification task. We focus on fine-tuning a classification layer implemented as a standard feed-forward and a softmax layer on top of the pre-trained BERT.

For fine-tuning the target model, we keep the hyper-parameters the same as in the pre-training by Devlin [31], except for the batch size, learning rate, and number of training epochs. The optimal hyper-parameter values are task-specific. Devlin et al. found the following range of possible values to work well across different tasks [31]:

- Batch size: 32, 64
- Learning rate: 5e-5, 3e-5, 2e-5
- Number of epochs: 2, 3, 4

For our model, we decide to choose a small learning rate, and train with a few epochs. Because most of transfer learning models suffer from the so-called catastrophic forgetting problem. That is, the learnt information is erased when learning the new knowledge from the target domain data. Using an aggressive learning rate such as 6e-4 makes the training fail to converge. The BERT authors [31] recommend a number of training epochs between 2 and 4. Selecting a large number of epochs may cause over-fitting. As shown in Table 11,

we fine-tune our model for 2 epochs with a learning rate of 4e-5. A larger batch size often results in lower accuracy but faster epochs. In order to find the optimum batch size we perform several runs of varying batch sizes while keeping other parameters constant.

4.6 DeepEmotex: Experimental Results

After developing DeepEmotex models using USE and BERT, we run several experiments to fine-tune proposed models based on the emotion-classification task using our input dataset.

4.6.1 DeepEmotex: Emotion Dataset

To be able to use deep learning for modeling emotion, we needed a large dataset of labeled tweets. Since there are not many human-labeled datasets publicly available, we collect tweets with emotion-carrying hashtags as a surrogate for emotion labels [43, 44]. We defined four emotion classes (i.e., joy, relax, anger, and sadness), based on the emotion model proposed by Circumplex model [107]. We only collect the tweets labeled with these emotions. The details of our data collection process is described in Section 3.2.1.

In order to collect enough tweets to serve as our labeled dataset, we developed a list of hashtags representing each of four emotion classes proposed by the Circumplex model. For each emotion class, we prepared a set of hashtags representing the emotion. We then used the set of hashtags to extract tweets with hashtags. We used Twitter API to crawl Twitter with hashtags.

One advantage of using hashtags for labeling emotion data, is that the label is assigned by the writer of the tweet rather than an annotator who could wrongly decide the category of a tweet. After all, emotion is a fuzzy concept. Another advantage of this method is obviously that it enables us to acquire a sufficiently large training set to apply fine-tuning in our transfer learning approach.

4.6.1.1 Pre-processing Collected Tweets

Twitter data is very noisy, not only because of use of non-standard typography (which is less of a problem here) but due to the many duplicate tweets and the fact that tweets often have multiple emotion hashtags. Since these reduce our ability to build accurate models, we clean the data. We first apply some general cleaning. We convert the Tweets to lower-case letters. We remove Non-ascii letters, urls, "@NAME" and duplicate letters. We also filter out all retweets based on existence of the token "RT" regardless of its case. Since our goal is to create non-overlapping categories at the level of a tweet, we also remove all tweets with hashtags belonging to more than one emotion of the four emotion

Class	Happy-Active	Happy-Inactive	Unhappy-Active	Unhappy-Inactive	Total
#Tweets	148571	195313	149287	47354	540525

Table 10: Number of tweets collected as labeled data

Parameters	Epoch	Batch sizes	Learning rate
Values	2	50,100,150,200,250	4e-5

Table 11: Parameters of fine-tuning BERT and ROBERTA

categories. Table 10 represents the number of labeled tweets selected for each class after pre-processing.

4.6.2 DeepEmotex: Experimental Results of Fine-tuning USE

We train our models with frozen and unfrozen layers for 20 epochs. The training is conducted on our collected tweets. Our dataset comprises a total of 540,525 tweets labeled with four emotion classes as described in Section 4.6.1. We first shuffle our labeled datasets and create train, validation and test datasets. We then train our models on a total of 300,000 tweets as our training dataset, validate on 60,000 tweets and use 180,525 tweets as our test dataset. The training data is used for fine-tuning and the testing dataset is used for evaluation.

Figure 25 shows the classification accuracy of our two models with frozen and unfrozen layers on the validation set in terms of micro-average F1 score. As it shows, the accuracy of our models stabilize at about epoch 14. The final validation results show that our model with frozen layers gets an accuracy of $90\% \pm 0.82\%$ (Mean $\pm$ Standard deviation) after training for 20 epochs. The frozen model gets an accuracy of 90.6% on our test dataset. This is consistent with our validation dataset.

The final validation results show that our model with unfrozen layers gets an accuracy of $90.6\% \pm 0.85\%$ (Mean $\pm$ Standard deviation) after training for 20 epochs. We get an accuracy of 91% on our test dataset using the model with unfrozen layers. Comparing the results of the two models with frozen and unfrozen layers using statistical t-test shows that the unfreezing approach performed better than the freezing approach ($p - value = 0.007$).

4.6.3 DeepEmotex: Experimental Results of Fine-tuning BERT

The fine-tuning our BERT model is conducted on collected tweets. Our dataset comprises a total of 540,525 tweets labeled with four emotion classes as described in Section 4.6.1. The collected tweets are shuffled and divided into training, validation and test datasets.

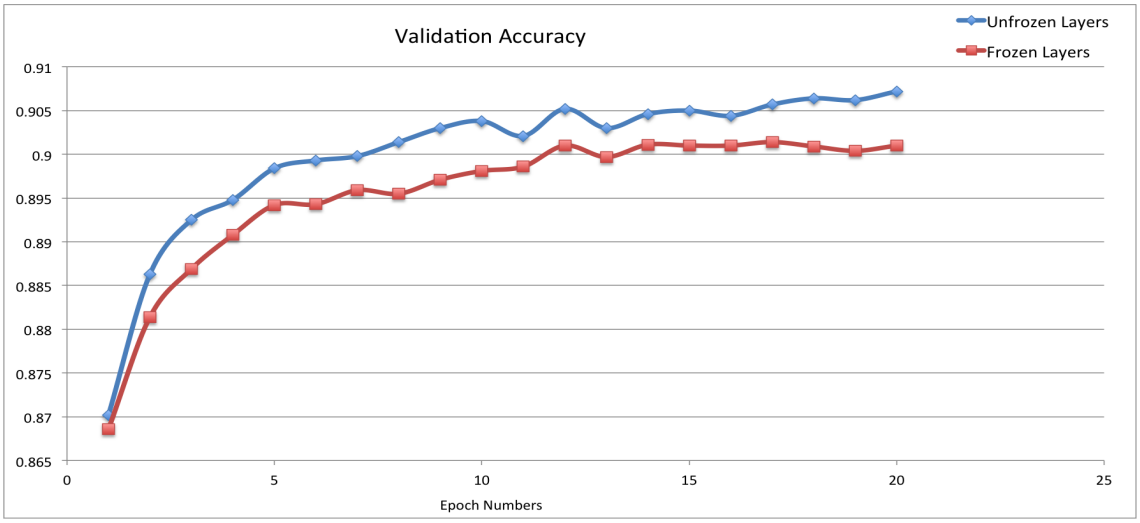

Figure 25: Emotion Classification Accuracy of fine-tuning USE with frozen and unfrozen layers on validation set in terms of micro-average F1 score.

We train our models on a total of 300,000 tweets as our training dataset, validate on 60,000 tweets and use 180,525 tweets as our test dataset. The training data is used for fine-tuning and the testing dataset is used for evaluation. The objective is to correctly predict the emotion of each tweet.

We fine-tune our BERT model using different batch sizes for 2 epochs. Figure 26 shows classification accuracy of DeepEmotex models created using different batch sizes to fine-tune BERT. As it shows the model achieved an accuracy of $91.8\% \pm 0.2\%$ on our test data in terms of MCC (Matthews correlation coefficient) score. The highest accuracy 92.1% is achieved when the batch size is 150.

MCC is a valid and precise metric to measure classification accuracy. Accuracy can show overoptimistic inflated results on imbalanced datasets. For balanced datasets, accuracy and MCC are synonymous [19, 24]. MCC is calculated as the following:

$$MCC = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP}) \times (\text{TP} + \text{FN}) \times (\text{TN} + \text{FP}) \times (\text{TN} + \text{FN})}} \tag{11}$$

4.6.4 Evaluating DeepEmotex

We evaluate the performance of the DeepEmotex models on emotion classification using benchmark datasets.

An issue with many of the benchmark datasets is data scarcity, which is specially problematic in emotion analysis [36].

To evaluate our DeepEmotex method on emotion detection we make use of EmoInt

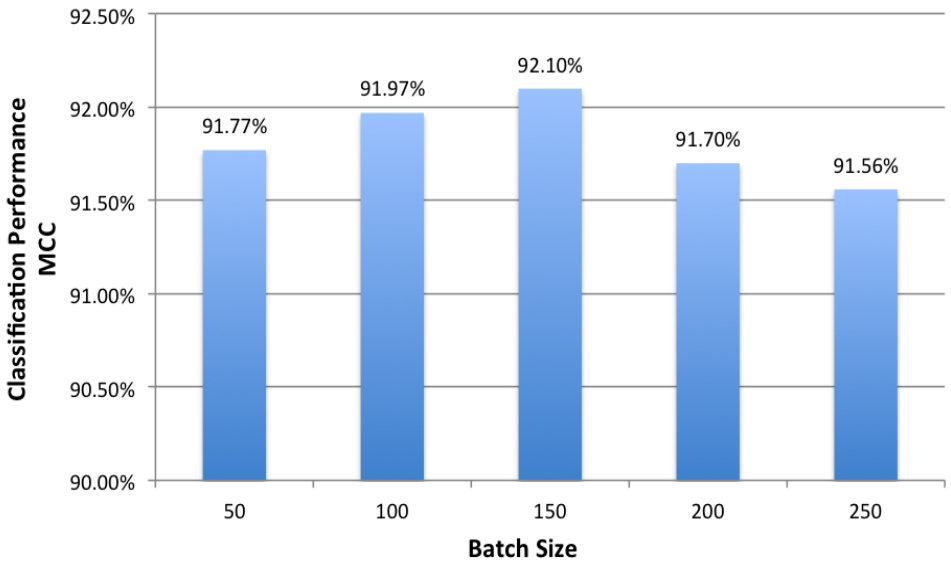

Figure 26: Classification results of fine-tuning BERT on test dataset using different batch sizes

Emotion Class	Joy	Sadness	Anger
Number of messages	671	656	641

Table 12: Number of messages in each emotion class in EmoInt benchmark dataset

as a benchmark dataset. EmoInt (Emotion Intensity in Tweets) is a dataset of emotions in tweets from WASSA 2017 [83]. The benchmark dataset provides four emotion classes including joy, anger, fear, and sadness. We only evaluate the emotion classes happy-active, unhappy-active, and unhappy-inactive as the happy-inactive emotion is not provided in the benchmark dataset.

Table 12 lists the number of samples in each class in EmoInt benchmark dataset. As it shows, there are about equal number of samples belonging to each class.

For evaluating our models with the benchmark dataset, we associate joy with happy-active, anger with unhappy-active, and sadness with unhappy-inactive. Our fine-tuned models are utilized to classify the benchmark dataset. The models classify the input benchmark dataset into our four emotion classes. The input class joy is correctly classified if it is labeled as happy-active or happy-inactive. The input class anger is correctly classified if it is labeled as unhappy-active and the input class sadness is correctly classified if it is labeled as unhappy-inactive. We measure the classification accuracy of each class using the following definition:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{12}$$

Emotion Class \ Batch Size	BERT-50	BERT-100	BERT-150	BERT-200	BERT-250	Bi-LSTM-CNN
Joy	84%	**87%**	81%	86%	85%	65.8%
Anger	37%	40%	38%	**44%**	42%	66.6%
Sadness	**81%**	78%	80%	**81%**	72	70.6%
Overall	67%	68%	66%	**70%**	66%	67.6%

Table 13: Classification accuracy of DeepEmotex using BERT to predict emotion in the benchmark datasets

Table 13 presents the evaluation results of DeepEmotex's models created using different batch sizes to fine-tune BERT. As it shows, the anger class achieved the lowest accuracy among all the emotion classes. The joy class achieved the highest accuracy among all the emotion classes. Among different batch sizes, the highest accuracy in sadness is achieved by the models created using batch size 50 and 200. Also, the model created by using the batch size 100 achieved the highest accuracy in joy emotion class. The model created using the batch size 200 achieved the highest accuracy over all emotion classes.

Table 14 lists the evaluation results of DeepEmotex models created using Universal Sentence Encoder by freezing and unfreezing the layers. As the table presents the model created by unfreezing layers achieved higher classification accuracy than the model created by freezing layers.

Comparing the evaluation results of Table 13 with Table 14 shows that the models created using BERT achieved a higher performance than the models created using USE in classifying emotion in the EmoInt benchmark dataset. The USE models were able to correctly classify emotion in 58% of the EmoInt benchmark dataset. Our BERT models achieved 70% accuracy in classifying emotion in the EmoInt dataset. Evaluation results show that the proposed BERT model outperformed the Bi-LSTM-CNN model [48] for joy and sadness classes.

4.7 Conclusion

In this research, we study deep learning methods to detect emotion in text messages. We develop and evaluate deep learning models, called DeepEmotex, to automatically classify emotion in text messages. DeepEmotex models automatically learn multiple layers of feature representations for emotion classification and thus reduce the need for hand-crafted features.

DeepEmotex utilizes sequential transfer learning to apply the knowledge learned from

Emotion Class	USE-Freezing layers	USE-Unfreezing layers	Bi-LSTM-CNN
Joy	**70%**	**70%**	65.8%
Anger	50%	**52%**	66.6%
Sadness	51%	**52%**	70.6%
Overall	57%	**58%**	67.6%

Table 14: Classification accuracy of DeepEmotex using USE to predict emotion in the benchmark dataset

pre-trained models on large text corpus. DeepEmotex uses Universal Sentence Encoders and BERT as pre-trained models. These models are fine-tuned to utilize them for our emotion classification task and obtain state-of-the-art results.

Experimental results and analysis demonstrate the effectiveness of the fine-tuned models. Created models were able to achieve over 91% accuracy for multi-class emotion classification on test dataset. DeepEmotex models achieved a higher classification accuracy than Emotex by more than 2% (See Section 3.3.2).

We also evaluated the performance of the models created using BERT and the models created using USE in classifying emotion in the EmoInt benchmark dataset. The BERT's models were able to correctly classify emotion in 70% of the instances in the benchmark dataset. Our evaluation results shows that the proposed BERT model outperformed the Bi-LSTM-CNN model [48] by about 3%.

5 EmotexStream: A Framework for Analyzing Emotion in Live Streams of Text Messages

5.1 Introduction

After developing emotion classification models, we now aim to deploy the trained models to analyze emotion in live streams of tweets. However analyzing text in real time is challenging due to the noise and fast-paced nature of tweets in the wild. For this, we develop a two-stage approach for classifying live streams of tweets.

Twitter messages cover a wide range of subjects. However, since our focus is on emotion detection, we are only interested in processing messages that contain emotions. For instance, the tweet "I have a wonderful roommate" conveys a happy emotion and is a good input to our system. In contrast, the tweet "It's time for bed" cannot be identified as expressing any type of emotion neither happy nor sad. Therefore we aim to identify such tweets without emotion and eliminate them in a fast pre-classification step. In fact, we decompose the emotion detection task into two sub-tasks. We first detect tweets without any identifiable emotion using a binary classifier. Then we conduct a fine-grained emotion classification on tweets with explicit emotion.

Figure 27 shows our emotion analysis pipeline in classifying the general stream of tweets [45, 46]. As it shows, after cleaning and preprocessing of tweets we categorize tweets into two general classes, namely emotion-present and emotion-absent tweets. For binary classification of tweets we develop an unsupervised method that utilizes emotion lexicons. Our binary classifier assumes that tweets with no emotion are the ones without any emotional or affective words. Therefore, it classifies tweets containing at least one affective or emotional word as emotion-present tweets, and classifies tweets without any affective word as emotion-absent tweets. As we described in Section 3.2.3.1, different emotion lexicons are available, including ANEW lexicon, LIWC dictionary, and AFINN. We utilize all the affective words from these three lexicons and create a comprehensive affective lexicon for our binary classification task.

After binary classification, emotion tweets will then go through the feature selection and multi-class emotion classifier generated by our Emotex technology to classify them based on our defined classes of emotion.

We develop DeepEmotex models after EmotexStream system. Thus, DeepEmotex models are not deployed to classify live streams of tweets using EmotexStream framework.

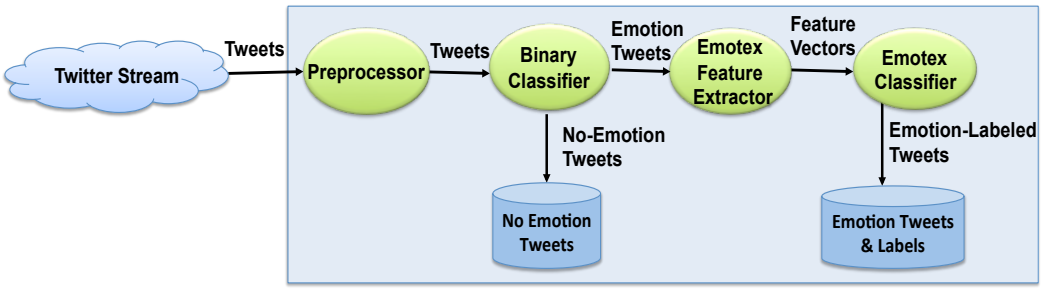

Figure 27: EmotexStream: A two-stage approach to classify live streams of tweets

5.2 Proposed Approach to Detect Emotion-Intensive Moments in Live Streams of Messages

Detecting and measuring emotion in social networks such as Twitter enable us to observe crowd emotion and behavior. Using EmotexStream we are able to classify live streams of tweets. We now aim to use our EmotexStream system to measure public emotion and detect emotion-burst moments in live stream of tweets. We are looking for the percentage of people in a geographic location experiencing certain emotions during a specific time. The goal is to explore temporal distributions of aggregate emotion and detect temporal bursts in public emotion from live text streams.

For this purpose, we first apply our EmotexStream system to automatically detect the emotion of people from their messages in live stream of tweets. As shown in Figure 28 EmotexStream converts live text streams into streams of emotion classes. Then we aggregate the emotion stream of each class into a time-based histogram to analyze public emotion trends and discover emotion-evolving patterns over time. We propose an online method to measure public emotion and detect abrupt changes in emotion as emotion-intensive moments in live text streams [46]. Before describing our online method to detect important moments in social streams, we define some concepts in the context of tweet streams as below:

Definition 4 (Emotion Stream). *An emotion stream S_E is a continuous sequence of time-ordered messages $M_1, M_2, \cdots M_r, \cdots$ from a tweet stream, such that each message M_i belongs to a specific emotion class $E_{c1} \in E_{Class}$ (E_{Class} is the set of predefined emotion classes defined in Section 3).*

In order to estimate the value of a specific emotion class E_{c1} among the people in a geographic location L during a time period $[T_1, T_2]$, we define a function as below:

$$E_{public}(T_1, T_2, L, E_{c1}) = \sum_{T_1 < T_i < T_2,\ L_i \in L} F(M_i, E_{c1}) \tag{13}$$

where $M_i =< U_i, T_i, L_i, C_i, E_i >$ is a tweet message in the emotion stream from the emo-

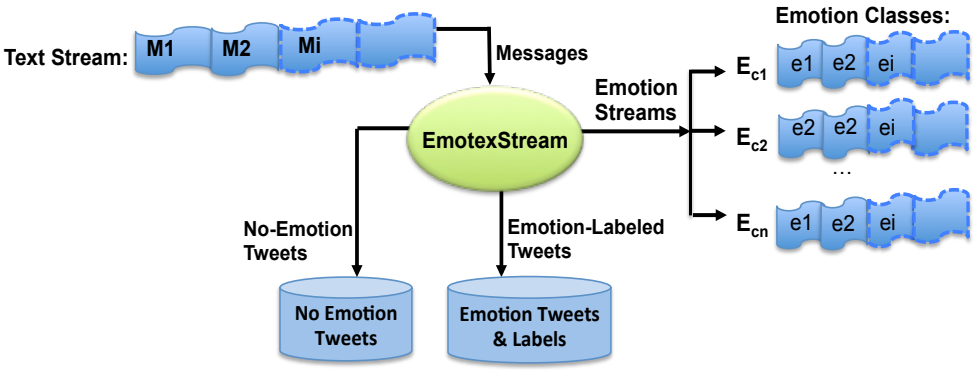

Figure 28: Converting text streams into emotion streams using EmotexStream

tion class $E_i \in E_{Class}$, posted by user U_i in location $L_i \in L$, at the time $T_1 < T_i < T_2$, and $F(M_i, E_{c1})$ is an indicator function defined as below:

$$F(M_i, E_{c1}) = \begin{cases} 1 & \text{if } M_i \in E_{c1}, \\ 0 & \text{Otherwise.} \end{cases} \tag{14}$$

Using equation 13 we can quantify emotion of a population in a geographic location and during a time period. We can then analyze such emotion streams to detect temporal bursts of crowd emotion. These sudden bursts are characterized by a change in the fractional presence of messages in particular emotion classes. Formally, we define such abrupt changes as "emotion burst", which can point towards important moments. In order to detect emotion bursts, we determine the higher or the lower rate at which messages have arrived to an emotion class in the current time window of length W. Two parameters α and β are used to measure this evolution rate.

Definition 5 (Emotion Burst). *An emotion burst over a temporal window of length W at the current time T_c is said to have occurred in a geographic region L, if the presence of a specific class emotion E_{c1} during a time period $(T_c - W, T_c)$ is less than the lower threshold α or greater than the upper threshold β.*

In other words, we should have either

$$E_{public}(T_c - W, T_c, L, E_{c1}) \leq \alpha \tag{15}$$

or

$$E_{public}(T_c - W, T_c, L, E_{c1}) \geq \beta. \tag{16}$$

Now we need to define the upper bound α and lower bound β of public emotion for each emotion class during a temporal window. If our algorithm is applied offline (i.e.

all the tweets are available), the thresholds can be estimated from the average sum over the whole time period. However in the online approach all the tweets are not available. Therefore, in the online approach, we compute the thresholds from the tweets in a temporal sliding window, where the size of the moving window is a parameter.

Figure 29 presents our system for detecting important moments in live text streams. Emotion streams can be created by applying EmotexStream system to classify tweets arriving in a stream. Let $e_1, \cdots e_i, \cdots e_n$ denote the emotion values of class E_{c1} of the tweets posted within a temporal window of length W in an emotion stream (n is the number of tweets posted within W). Apparently, $e_1, \cdots e_i, \cdots e_n$ are independent 0-1 random variables (e_i=0 means message M_i doesn't belong to the emotion class E_{c1}, and e_i=1 means message M_i belongs to the emotion class E_{c1}). Emotion aggregator uses Equation 13 to measure public emotion over a period of time. Based on Equation 13, public emotion within the temporal window W is defined as below:

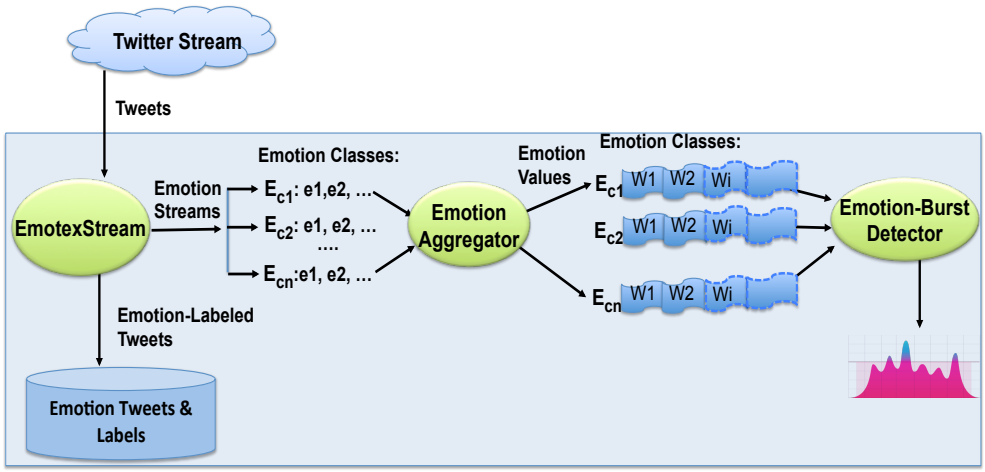

Figure 29: Detecting emotion-bursts in live text streams

$$E_{public}(T_c - W, T_c, L, E_{c1}) = \sum_{i=1...n} F(M_i, E_{c1}) \tag{17}$$

where $F(M_i, E_{c1})$ is an indicator function of E_{c1} and n is the number of tweets posted within W. As we know Hoeffding's inequality provides an upper bound on the probability that the sum of random variables deviates $\lambda > 0$ from its expected value as shown by Equation 18:

$$Pr[|X - \mu| >= \lambda] <= 2e^{-2\lambda^2/n} \tag{18}$$

where X is the sum of independent random variables $X_1, X_2, \cdots, X_n$, with $E[X_i] = p_i$, and the expected value $E[X] = \sum_{i=1...n} p_i = \mu$.

According to the Central Limit Theorem, if n is large then X approaches a normal distribution. We can use Hoeffding's inequality to define an upper bound on the probability that the public emotion E_{c1} deviates from its expected value. Using the Hoeffding bound, for any $\lambda > 0$ we have:

$$Pr[|E_{public}(T_c - W, T_c, L, E_{c1}) - \mu_e| >= \lambda] <= 2e^{-2\lambda^2/n} \quad (19)$$

where μ_e is the expected number of tweets belong to the emotion class E_{c1} in window W and n is the number of tweets posted within W. Given that n is large in a Tweet Stream, emotion class E_{c1} can be approximated using a normal distribution.

$$\mu_e = n \times P_e$$

where P_e is the expected rate of the emotion class E_{c1}.

We use the historical average rate of each emotion class as expected rate P_e for that emotion class. For example, a weekly window can be used to average the rate of each emotion class based on all tweets in general. Therefore, other than a sliding detection window over the recent tweets posted about a topic, we also utilize a larger reference window to summarize the information about the tweets posted in general. In fact, our emotion-burst detection methodology utilizes two sliding windows. One small window that keeps the rate of each emotion class based on the most recent tweets posted about a topic. Another large reference window that keeps the average rate of each emotion class based on all the past tweets posted in general.

Now we describe our methodology to automatically discover emotion bursts during a real life event. First, we create an emotion stream by applying the model created by Emotex system to classify tweets arriving in a stream based on a predefined set of emotion classes. As a second step, our emotion burst detection algorithm then aggregates the tweets of each emotion class into a time-based histogram, using the function in Equation 13. This aggregation allows us to count the rate of each emotion class in each time period. We then define a sliding window W_{topic} (e.g., daily) over the stream of tweets about a topic aggregated in temporal bins. We also define a large (e.g., weekly) window $W_{general}$ over the general stream of tweets to keep track of the average rate of each emotion class. In order to perform the burst detection, we continuously monitor the rate of public emotion for each emotion class within each temporal window W_{topic}. Whenever the rate of an emotion class exceeds the upper threshold β or falls beneath the lower limit α, an emotion burst is marked as an important moment by keeping its time of occurrence and if it is an up or down case. Then the system signals the occurrence of the detected moments.

5.3 EmotexStream Experimental Results

During this experiment, we apply our emotion classifiers to classify the live streams of tweets using EmotexStream system (see Section 5.1). In this experiment we also select a real-life event and detect the emotion-intensive moments using our method described in Section 5.2. The following sections describe more details about each experiment.

5.3.1 Classifying Emotion in Live Streams of Tweets

To classify emotion in live streams of tweets, we utilize EmotexStream framework presented in Section 5.1. Based on the EmotexStream system, we first detect emotion-present tweets and separate them from emotion-absent tweets. Therefore, we utilize our binary classifier developed using several emotion lexicons. For the binary classification experiment we collect a large amount of general tweets from United States without filtering them by any specific hashtag or keyword (see Table 15). After cleaning up the noise, we classify them using our binary classifier. Emotion-present tweets will then go through the feature selection and multi-class emotion classifier generated by our Emotex system to classify them based on our defined classes of emotion. Table 15 shows the results of our binary classification experiment. It is interesting to observe that in a random sample of tweets the majority does in fact contain identifiable emotion.

We also evaluate the accuracy of our binary classifier through a user study. We randomly select a sample set of general tweets including 50 tweets from the dataset described in Table 15. Then we ask 25 graduate students to manually classify them. They classified each tweet into two groups namely, emotion-present tweets versus emotion-free tweets (i.e., tweets with explicit emotion versus tweets without any emotion). Fleiss-Kappa for the labelers is 0.28 which shows a fair agreement. The manual label of each sample tweet is selected based on the majority votes of labelers for that tweet. There were three tweets which didn't receive the absolute majority of the votes. We didn't consider them in our evaluation. After creating manual labels, we classified the selected sample tweets using our binary classifier and compared them with manually classified results. The manual labels served as the ground truth labels. We generated our binary classifier results using two different lexicons LIWC and ANEW.

Table 16 shows the precision, recall and F-measure ($\beta = 1$) of the binary classifier through comparison with the manual classification. As the results show, using a larger lexicon (i.e., LIWC and ANEW) increased recall and F-measure, compared with using only one lexicon. Therefore for the binary classification task we use a multi-lexicon by combining different lexicons. A larger lexicon increases recall, but may decrease precision.

	Total Tweets	After pre-processing	Emotion Tweets	No-emotion Tweets
Number	105,134	104,924	56,472	48,452
Percent	100%	99.8%	53.7%	46.1%

Table 15: Results of binary classification in live stream of tweets

Lexicon	Precision	Recall	F-Measure
LIWC	93%	77%	84.3%
ANEW	74%	82%	77.8%
LIWC& ANEW	78%	95%	85.6%

Table 16: Evaluating binary classification results by comparing them with manually classification results

5.3.2 Case Study: Detecting Emotion-bursts in Live Tweet Streams

Using EmotexStream we are able to classify live streams of tweets in real-time. We now use this system to measure and analyze public emotion in a specific location. The objective of this experiment is to observe the temporal distribution of crowd emotion and detect important moments during the real-life events. We select the death of Eric Garner in New York [4] which stirred public protests and rallies with charges of police brutality. Eric Garner died after a police officer put him in a choke-hold, which caused many discussions on social media. On December 3, 2014, a grand jury decided not to indict the police officer. We utilize the Twitter search API to search for tweets containing a specified set of hashtags. We collected 4K tweets containing the hashtag "Garner" from November 24 2015 until January 5 2015 posted in New York. After collecting tweets we classify them using our EmotexStream model (see Section 5.1). Then, the emotion-classified tweets are aggregated into a daily-based histogram. Finally, using the methodology described in Section 5.2 we measure public emotion and detect emotion-critical moments.

Figure 30 presents the temporal changes of different classes of emotion in New York during the selected event. The important moments are also specified in this figure. The distribution shows a predominance of sad and angry emotions over happy emotion in many days during the event. In order to predict the important moments as emotion bursts, we apply a sliding window W_{event} of length one day over the emotion stream of tweets aggregated in daily bins, as described in Section 5.2. Also a reference weekly window $W_{general}$ is applied over the general stream of tweets to calculate the average rate of each emotion class. Then, we continuously monitor the frequency rate $E_{public}(Tc - W_{event}, Tc, L, E_{c1})$ over time for each emotion class E_{c1}. Whenever this rate for an emotion class exceeds the

[4] https://en.wikipedia.org/wiki/Death_of_Eric_Garner

upper threshold of β or falls beneath the lower limit α, an emotion burst is reported.

Table 17 presents the days of abrupt changes in happiness. The second row shows the frequency rate of emotion bursts which are out of range. The last row shows the low and high boundaries. Comparing the results of this table with the important moments specified in Figure 30 confirms that our method is able to detect emotion-critical moments.

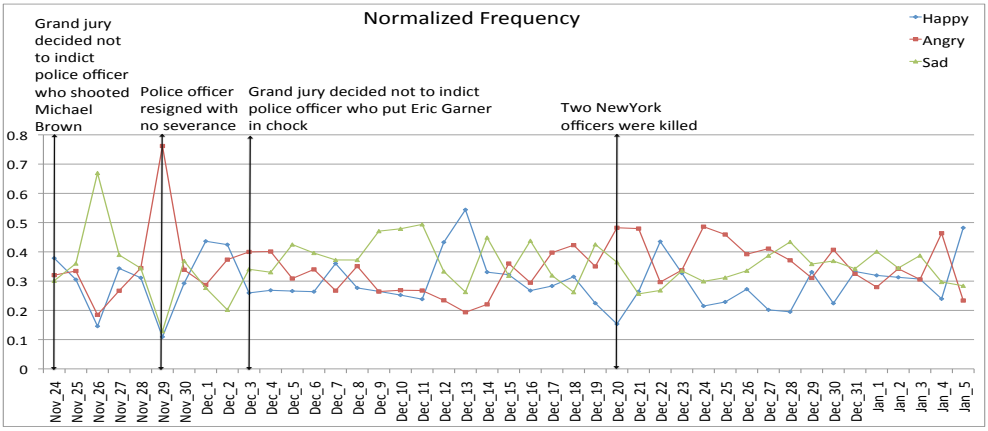

Figure 30: Changes of emotion about selected sad events in New York

Date	Nov 26	Nov 29	Dec 19	Dec 20	Dec 27	Dec 28	Dec 30
Happy Rate	210	175	576	462	463	360	503
Boundary (α, β)	(360, 936)	(400, 1040)	(641, 1668)	(753, 1957)	(573, 1491)	(461, 1199)	(561, 1459)

Table 17: Detected burst changes in happiness

5.4 Conclusion

We develop a two-stage framework called EmotexStream to classify live streams of tweets. First it separates tweets with explicit emotion from tweets without any emotion using a binary classifier. Then it utilizes our emotion classification models to classify the tweets with explicit emotion into fine-grained emotion classes. Moreover, we propose an online method to measure public emotion and detect emotion-intensive moments in streams of text messages which can be used for online emotion tracking.

Finally, we evaluate EmotexStream framework by running some case studies. To evaluate EmotexStream, we deploy it to measure crowd emotion and detect emotion-intensive moments during several real-life events in some cities in US. The evaluation results show that our proposed method is able to detect emotion-intensive moments during the selected

events.

6 Conclusion and Future Directions

In this dissertation, we have made contributions to the area of emotion analysis in natural language messages.

This chapter concludes the work described in this dissertation. A brief summary of contributions and the research impact are presented in Sections 6.1 and 6.2. Avenues of future work are discussed in Section 6.3.

6.1 Contributions

In this research we first utilized linear supervised learning to detect emotion in texts, and the idea of considering Twitter hashtags as automatic emotion labels [44, 45]. We validated the effectiveness of utilizing our hashtag-based labeling concept through two user studies, one with psychology experts and the other with the general crowd [43].

We then extend our Emotex system and develop a framework, called EmotexStream [45, 46] to analyze emotion in live streams of text messages. We propose an online method to measure public emotion and detect emotion-intensive moments. To evaluate EmotexStream, we deploy it to measure public emotion and investigate its temporal distribution during several real-life events in a geographic location (e.g., a city). The evaluation results show that our proposed method is able to detect emotion-intensive moments during the events.

Furthermore, we studied deep learning models to classify emotion in texts. We utilized transfer learning to fine-tune the pre-trained models. We presented and evaluated different methods for transfer learning scenarios. We utilized state-of-the-art pre-trained models including USE and BERT and fine-tune them for our emotion classification task using our domain-specific emotion datasets. Experimental results and analysis demonstrate the effectiveness of the fine-tuned models. Created models were able to achieve over 91% accuracy for multi-class emotion classification on test dataset.

In particular, we made the following major contributions in this dissertation:

- Overcoming the vague boundaries of emotion classes: We address this issue using a two-pronged approach. First, we define the emotion classes based on the Circumplex model of affect [16]. Instead of a small number of discrete categories, this model defines the emotion in terms of latent dimensions (e.g., arousal and valence). Second, a soft classification approach is proposed to measure the probability of assigning a message into each emotion class, in addition to a typical classification that simply assigns one single emotion class to each text message in a deterministic manner.

- Proposed a distant supervision method to collect labeled data: We conjecture that emotional hashtags inserted by authors indicate the main emotion expressed by their Twitter message. This approach overcomes the need for manual labeling and yields a completely automatic scheme to obtain large amounts of labeled data. This strategy could equally be applied in other applications where labeling is required to automatically obtain large amounts of supervised data.

- Emotion detection using supervised machine learning: We develop the Emotex system to automatically classify emotion expressed in text messages. Emotex uses sparse one-hot model to represent feature vectors. The features are selected based on a predefined set of emotion lexicons. We train emotion classification models and report their soft and hard classification results. We evaluate the classification accuracy of Emotex by comparing it with the lexical approach. Classification accuracy of Emotex is about 90%, while the accuracy of the lexical approach is about 66%.

- Emotion detection using neural transfer learning: We develop a deep learning framework called DeepEmotex to classify emotion in text messages. DeepEmotex learns emotion-specific features based on the input textual context using sequential transfer learning. For this, DeepEmotex develops methods for fine-tuning the pre-trained language models to learn emotion-specific features. We analyze the adaptation or fine-tuning phase during which the pre-trained knowledge is transferred to our emotion classification task. Using the state-of-the-art pre-trained models, we achieve 92% classification accuracy on our test dataset. DeepEmotex models achieved a higher classification accuracy than Emotex by more than 2%.

  We also evaluate the performance of DeepEmotex models in classifying emotion in the benchmark datasets. DeepEmotex models were able to correctly classify emotion in 70% of the samples in EmoInt benchmark dataset.

- Emotion-burst detection during social events: We develop EmotexSream framework to classify live streams of tweets. We propose an online method to measure public emotion and detect emotion-intensive moments. Our method can be used for online emotion tracking during social events. We evaluate EmotexStream framework by conducting several case studies using live and unfiltered streams of tweets during real-life events.

6.2 Impact of Research

Social networks such as Twitter provide valuable information to observe crowd emotion and behavior and study a variety of human behavior and characteristics [123]. Such net-

works are appropriate data sources for studying the emotions of individuals as well as larger populations. Interesting applications of behavioral studies include detecting mood after a disaster, analyzing political mood, or understanding emotion about certain products.

Increasing evidence suggests that emotion detection and screening will be effective in many health applications [25, 89, 41, 105]. The development of robust textual emotion sensing technologies promises to have a substantial impact on public and individual health and urban planning. Such emotion mining tools, once available, could potentially be employed in a large variety of applications ranging from population level studies of emotions, the provision of mental health counseling services over social media, and other emotion management applications.

The census bureau and other polling organizations may be able to use the emotion mining technology to estimate the percentage of people in a community experiencing certain emotions and correlate this with current events and various other aspects of urban living conditions. This type of technology can also enhance early outbreak warning for public health authorities so that a rapid action can take place [56].

The emotion mining tools could also be used by counseling agencies to monitor emotional states of individuals or to recognize anxiety or systemic stressors of populations [43]. For instance, university counseling centers could be warned early about distressed students that may require further personal assessment.

Moreover, studying public emotion promises to be of great value in several fields from social science, political science, public health research to market research, that are interested in aggregate emotion instead of individual cases. It could assist government agencies in recognizing growing public fear or anger associated with a particular decision or event or in helping them to understand the public's emotional response toward controversial issues or international affairs. In some cases rapidly gaining such insights as well as getting a deeper understanding on trends associated with positive versus negative emotion propagation across a population can be critical.

The analysis of emotion during real-life events helps to realize the public emotion regarding the event. Important events are often discussed widely in social networks. Public emotion analysis can aid public health researchers by providing them with (1) a low-cost method to detect emotion-critical events across different sub-populations; (2) useful knowledge for identifying at-risk populations; and (3) a method to formulate new hypotheses about the impact of real-time events on populations.

6.3 Future Directions

The work presented in this dissertation outlines emotion detection in text. While the algorithms presented in this research represent significant progress in detecting emotion in text documents, there are still many interesting research directions that deserve further pursuit.

1. Sarcasm detection: Researchers on sarcasm detection work hard to identify sarcastic comments with high accuracy, as human emotions and attitudes are often ambiguous [2]. It is difficult for the machine to identify sarcastic statements. Sarcasm detection is faced by the problem of limited-sized labeled training data. The problem of constructing manually labeled sarcasm datasets is even far more severe as sarcasm is a remarkably rare positive class [2]. This makes sarcasm detection challenging for supervised learning methods.

2. Mental disorder detection: An interesting application of this research is to identify mental disorders. Social networks contain a large corpus of textual data that is rich with emotional content, which can be mined for a variety of purposes. Such networks are appropriate data sources for behavioral studies, especially for studying emotions of individuals as well as larger populations. Therefore, social networks such as Twitter provide valuable information to observe crowd emotion and behavior and study a variety of human behavior and characteristics [123].

3. Enhanced embeddings: One possible way of further enhancing the text representations, or specifically word embeddings, includes exploring dynamic embeddings, where the learned embeddings can be tweaked or updated with respect to evolving connotation of words over time [132]. Specifically, each word in a different time frame (e.g., years) is represented by a different vector. For example, apple which was traditionally only associated with fruits, is now also associated with a technology company. For this reason, understanding and tracking word evolution is useful for time-aware knowledge extraction tasks (e.g., public emotion analysis), and other applications in text mining. To this end, word embeddings can be learned with a temporal bent, for capturing time-aware meanings of words.

   Yet another avenue of future work could explore topic-enhanced word embeddings [67], where topic information is incorporated along with emotion information in order to generate more coherent text representations. The basic idea of topic-aware word embeddings is that, we allow each word to have different embeddings under different topics. For example, the word apple indicates a fruit under the topic food, and indicates a company under the topic information technology (IT).

4. Multi-Source domain adaptation: With the advent of multiple emotion detection corpora, multi-source domain adaptation provides an interesting future research direction. Multi-source domain adaptation is a setting where data from not one but multiple source domains are available for training. By leveraging the variability from multiple data sources collected under different contexts, multi-source emotion learning might be able to provide robust and global systems. Multi-source emotion learning can also lay a foundation for modeling various aspects of human behavior other than emotion (e.g., mood, anxiety), where only a limited number of datasets with a small number of data samples are available. Multi-source domain adaptation has not been explored for automatic emotion detection task, which makes it a great future research direction.

5. Unifying emotion labels in different domains: A potential challenge in automatic emotion recognition lies in the fact that the various datasets might include different types of emotional labels. This challenge can introduce high mismatch among different domains. Understanding and modeling associations between cross-domain emotions can potentially contribute toward more accurate emotion inferences in real-life. Synchronizing the various spectrum of emotion may provide additional benefits, which we leave for future exploration.

6. Cross-Cultural and Cross-Linguistic emotion detection: Emotions can be expressed in different ways across different cultures and languages. The relation of a word to emotion concepts may depend on cultural aspects that can be inferred from extensive word usage rather than from what can be found in dictionaries. Cross-language computational studies are challenging, compelling, and they can have an important practical value, for example when addressing topics such as emotions, negotiation and conflict. It would be beneficial to examine potential discrepancies related to the linguistic style for building global emotion recognition systems across cultures.

References

[1] M. Abdul-Mageed and L. Ungar. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 718–728, 2017.

[2] G. Abercrombie and D. Hovy. Putting sarcasm detection into context: The effects of class imbalance and manual labelling on supervised machine classification of twitter conversations. In *Proceedings of the ACL 2016 student research workshop*, pages 107–113, 2016.

[3] C. C. Aggarwal and C. Zhai. *Mining text data*. Springer Science & Business Media, 2012.

[4] A. Agrawal and A. An. Unsupervised emotion detection from text using semantic and syntactic relations. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 346–353. IEEE Computer Society, 2012.

[5] H. Al-Omari, M. A. Abdullah, and S. Shaikh. Emodet2: Emotion detection in english textual dialogue using bert and bilstm models. In *2020 11th International Conference on Information and Communication Systems (ICICS)*, pages 226–232. IEEE, 2020.

[6] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018.

[7] S. Aman and S. Szpakowicz. Identifying expressions of emotion in text. In *International Conference on Text, Speech and Dialogue*, pages 196–205. Springer, 2007.

[8] A. Arnold, R. Nallapati, and W. W. Cohen. A comparative study of methods for transductive transfer learning. In *ICDM Workshops*, pages 77–82, 2007.

[9] S. Arora, Y. Liang, and T. Ma. A simple but tough-to-beat baseline for sentence embeddings. 2016.

[10] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[11] A. Baevski, S. Edunov, Y. Liu, L. Zettlemoyer, and M. Auli. Cloze-driven pretraining of self-attention networks. *arXiv preprint arXiv:1903.07785*, 2019.

[12] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[13] L. Barbosa and J. Feng. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of the 23rd ACL: Posters*, pages 36–44. Association for Computational Linguistics, 2010.

[14] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[15] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[16] J. Bollen, H. Mao, and A. Pepe. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *ICWSM*, 11:450–453, 2011.

[17] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[18] M. M. Bradley and P. J. Lang. Affective norms for english words (anew): Instruction manual and affective ratings. Technical report, Citeseer, 1999.

[19] J. Brown. Classifiers and their metrics quantified. *Molecular informatics*, 37(1-2):1700127, 2018.

[20] R. A. Calvo and S. Mac Kim. Emotions in text: dimensional and categorical models. *Computational Intelligence*, 29(3):527–543, 2013.

[21] L. Canales, C. Strapparava, E. Boldrini, and P. Martnez-Barco. Exploiting a bootstrapping approach for automatic annotation of emotions in texts. In *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*, pages 726–734. IEEE, 2016.

[22] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al. Universal sentence encoder. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[23] A. Chatterjee, K. N. Narahari, M. Joshi, and P. Agrawal. Semeval-2019 task 3: Emocontext contextual emotion detection in text. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 39–48, 2019.

[24] D. Chicco and G. Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):6, 2020.

[25] M. D. Choudhury, M. Gamon, S. Counts, and E. Horvitz. Predicting depression via social media. In *ICWSM'13*. The AAAI Press, 2013.

[26] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[27] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[28] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[29] A. M. Dai and Q. V. Le. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087, 2015.

[30] M. De Choudhury, S. Counts, and M. Gamon. Not all moods are created equal! exploring human emotional states in social media. In *ICWSM'12*, 2012.

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

[32] P. S. Dodds and C. M. Danforth. Measuring the happiness of large-scale written expression: Songs, blogs, and presidents. *Journal of happiness studies*, 11(4):441–456, 2010.

[33] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[34] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*, 2019.

[35] P. Ekman. Basic emotions. *Handbook of cognition and emotion*, 98:45–60, 1999.

[36] B. Felbo, A. Mislove, A. Søgaard, I. Rahwan, and S. Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625, 2017.

[37] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12, 2009.

[38] Y. Goldberg and O. Levy. word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[39] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[40] H. Gunes, B. Schuller, M. Pantic, and R. Cowie. Emotion representation, analysis and synthesis in continuous space: A survey. In *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pages 827–834. IEEE, 2011.

[41] B. Guthier, R. Alharthi, R. Abaalkhail, and A. El Saddik. Detection and visualization of emotions in an affect-aware city. In *Proceedings of the 1st International Workshop on Emerging Multimedia Applications and Services for Smart Cities*, pages 23–28. ACM, 2014.

[42] M. U. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(Feb):307–361, 2012.

[43] M. Hasan, E. Agu, and E. Rundensteiner. Using hashtags as labels for supervised learning of emotions in twitter messages. In *Proceedings of the ACM SIGKDD Workshop on Healthcare Informatics, HI-KDD*, 2014.

[44] M. Hasan, E. Rundensteiner, and E. Agu. Emotex: Detecting emotions in twitter messages. In *Proceedings of the Sixth ASE International Conference on Social Computing (SocialCom 2014)*. Academy of Science and Engineering (ASE), USA, 2014.

[45] M. Hasan, E. Rundensteiner, and E. Agu. Automatic emotion detection in text streams by analyzing twitter data. *International Journal of Data Science and Analytics*, 7(1):35–51, 2019.

[46] M. Hasan, E. Rundensteiner, X. Kong, and E. Agu. Using social sensing to discover trends in public emotion. In *2017 IEEE 11th International Conference on Semantic Computing (ICSC)*, pages 172–179. IEEE, 2017.

[47] H. He, K. Gimpel, and J. Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586, 2015.

[48] Y. He, L.-C. Yu, K. R. Lai, and W. Liu. Yzu-nlp at emoint-2017: Determining emotion intensity using a bi-directional lstm-cnn model. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 238–242, 2017.

[49] F. Hill, K. Cho, and A. Korhonen. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*, 2016.

[50] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[51] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for nlp. *arXiv preprint arXiv:1902.00751*, 2019.

[52] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[53] B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050, 2014.

[54] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, 2015.

[55] T. Joachims. Making Large-Scale SVM Learning Practical. In B. Schölkopf, C. J. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, USA, 1999. MIT Press.

[56] N. Kanhabua and W. Nejdl. Understanding the diversity of tweets in the time of outbreaks. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 1335–1342. International World Wide Web Conferences Steering Committee, 2013.

[57] Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, 2014.

[58] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[59] M. Köper, E. Kim, and R. Klinger. Ims at emoint-2017: Emotion intensity prediction with affective norms, automatically extended resources and deep learning. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 50–57, 2017.

[60] E. Kouloumpis, T. Wilson, and J. Moore. Twitter sentiment analysis: The good the bad and the omg! In *ICWSM'11*. The AAAI Press, 2011.

[61] S. Lai, L. Xu, K. Liu, and J. Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273, 2015.

[62] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[63] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

[64] Y. Li, C.-Y. Chen, and W. W. Wasserman. Deep feature selection: Theory and application to identify enhancers and promoters. In *International Conference on Research in Computational Molecular Biology*, pages 205–217. Springer, 2015.

[65] Z. C. Lipton, J. Berkowitz, and C. Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[66] H. Liu, H. Lieberman, and T. Selker. A model of textual affect sensing using real-world knowledge. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 125–132. ACM, 2003.

[67] Y. Liu, Z. Liu, T.-S. Chua, and M. Sun. Topical word embeddings. In *Twenty-ninth AAAI conference on artificial intelligence*. Citeseer, 2015.

[68] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[69] L. Logeswaran and H. Lee. An efficient framework for learning sentence representations. *Sixth International Conference on Learning Representations (ICLR)*, 2018.

[70] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang. Transfer learning using computational intelligence: a survey. *Knowledge-Based Systems*, 80:14–23, 2015.

[71] L. Lucy and J. Gauthier. Are distributional representations ready for the real world? evaluating word vectors for grounded perceptual meaning. *arXiv preprint arXiv:1705.11168*, 2017.

[72] L. Luo and Y. Wang. Emotionx-hsu: Adopting pre-trained bert for emotion classification. *arXiv preprint arXiv:1907.09669*, 2019.

[73] C. Ma, H. Prendinger, and M. Ishizuka. Emotion estimation and reasoning based on affective textual interaction. In *Affective computing and intelligent interaction*, pages 622–628. Springer, 2005.

[74] B. McCann, J. Bradbury, C. Xiong, and R. Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305, 2017.

[75] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[76] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[77] T. Mikolov, S. Kombrink, L. Burget, J. Černockỳ, and S. Khudanpur. Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528–5531. IEEE, 2011.

[78] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[79] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of*

the Association for Computational Linguistics: Human Language Technologies, pages 746–751, 2013.

[80] A. Mnih and K. Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273, 2013.

[81] A. Mnih and Y. W. Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012.

[82] S. M. Mohammad. # emotional tweets. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 246–255. Association for Computational Linguistics, 2012.

[83] S. M. Mohammad and F. Bravo-Marquez. WASSA-2017 shared task on emotion intensity. In *Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA)*, Copenhagen, Denmark, 2017.

[84] A. Neviarouskaya, H. Prendinger, and M. Ishizuka. Textual affect sensing for sociable and expressive online communication. In *Affective Computing and Intelligent Interaction*, pages 218–229. Springer, 2007.

[85] F. A. Nielsen. A new anew: evaluation of a word list for sentiment analysis in microblogs. In *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages*, volume 718, pages 93–98, May 2011.

[86] M. Pagliardini, P. Gupta, and M. Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*, 2017.

[87] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. ELRA.

[88] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[89] M. Park, C. Cha, and M. Cha. Depressive moods of users portrayed in twitter. In *Proc. of the ACM SIGKDD Workshop on Healthcare Informatics, HI-KDD*, 2012.

[90] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

[91] J. W. Pennebaker, M. E. Francis, and R. J. Booth. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, page 71, 2001.

[92] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[93] M. Peters, S. Ruder, and N. A. Smith. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*, 2019.

[94] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[95] S. Poria, E. Cambria, and A. Gelbukh. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2539–2544, 2015.

[96] M. Purver and S. Battersby. Experimenting with distant supervision for emotion classification. In *Proceedings of the 13th EACL*, pages 482–491. Association for Computational Linguistics, 2012.

[97] A. Qadir and E. Riloff. Bootstrapped learning of emotion hashtags# hashtags4you. *WASSA 2013*, page 2, 2013.

[98] Q. Qian, M. Huang, J. Lei, and X. Zhu. Linguistically regularized lstms for sentiment classification. *arXiv preprint arXiv:1611.03949*, 2016.

[99] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[100] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[101] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[102] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[103] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[104] Y. Ren, Y. Zhang, M. Zhang, and D. Ji. Context-sensitive twitter sentiment classification using neural network. In *AAAI*, pages 215–221, 2016.

[105] B. Resch, A. Summa, P. Zeile, and M. Strube. Citizen-centric urban planning through extracting emotion information from twitter in an interdisciplinary space-time-linguistics algorithm. *Urban Planning*, 1(2):114–127, 2016.

[106] S. Ruder, M. E. Peters, S. Swayamdipta, and T. Wolf. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, 2019.

[107] J. A. Russell. A circumplex model of affect. *Journal of Personality and Social Psychology*, 39:1161–1178, 1980.

[108] J. A. Russell and L. F. Barrett. Core affect, prototypical emotional episodes, and other things called emotion: Dissecting the elephant. *Journal of personality and social psychology*, 76(5):805, 1999.

[109] H. Saif, Y. He, and H. Alani. Alleviating data sparsity for twitter sentiment analysis. CEUR Workshop Proceedings (CEUR-WS. org), 2012.

[110] A. Severyn and A. Moschitti. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 959–962. ACM, 2015.

[111] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[112] C. Strapparava and R. Mihalcea. Learning to identify emotions in text. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1556–1560. ACM, 2008.

[113] C. Strapparava and A. Valitutti. Wordnet affect: an affective extension of wordnet. In *Proceedings of 4th International Conference on Language Resources and Evaluation, LREC*, volume 4, pages 1083–1086, May 2004.

[114] S. Subramanian, A. Trischler, Y. Bengio, and C. J. Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*, 2018.

[115] M. Sundermeyer, R. Schlüter, and H. Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

[116] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[117] J. Suttles and N. Ide. Distant supervision for emotion classification with discrete binary values. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 121–136. Springer, 2013.

[118] K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *IN PROC. ACL*. Citeseer, 2015.

[119] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL (1)*, pages 1555–1565, 2014.

[120] Z. Teng, D. T. Vo, and Y. Zhang. Context-sensitive lexicon features for neural sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1629–1638, 2016.

[121] J. Turian, L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[122] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[123] S. Wakamiya, L. Belouaer, D. Brosset, R. Lee, Y. Kawai, K. Sumiya, and C. Claramunt. Measuring crowd mood in city space through twitter. In *International Symposium on Web and Wireless Geographical Information Systems*, pages 37–49. Springer, 2015.

[124] J. Wang, L.-C. Yu, K. R. Lai, and X. Zhang. Dimensional sentiment analysis using a regional cnn-lstm model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 225–230, 2016.

[125] W. Wang, L. Chen, K. Thirunarayan, and A. P. Sheth. Harnessing twitter big data for automatic emotion identification. In *2012 International Conference on Social Computing (SocialCom)*, pages 587–592. IEEE, 2012.

[126] X. Wang, F. Wei, X. Liu, M. Zhou, and M. Zhang. Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1031–1040. ACM, 2011.

[127] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358, 2015.

[128] A. Williams, N. Nangia, and S. R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

[129] H. Xu, B. Liu, L. Shu, and P. S. Yu. Bert post-training for review reading comprehension and aspect-based sentiment analysis. *arXiv preprint arXiv:1904.02232*, 2019.

[130] Q. Yang, Y. Zhang, W. Dai, and S. J. Pan. *Transfer learning*. Cambridge University Press, 2020.

[131] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

[132] Z. Yao, Y. Sun, W. Ding, N. Rao, and H. Xiong. Dynamic word embeddings for evolving semantic discovery. In *Proceedings of the eleventh acm international conference on web search and data mining*, pages 673–681, 2018.

[133] T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[134] L. Zhang, S. Wang, and B. Liu. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253, 2018.

[135] H. Zhao, Z. Lu, and P. Poupart. Self-adaptive hierarchical sentence model. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[136] C. Zhou, C. Sun, Z. Liu, and F. Lau. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*, 2015.

[137] S. Zhou, Q. Chen, and X. Wang. Active deep learning method for semi-supervised sentiment classification. *Neurocomputing*, 120:536–546, 2013.

[138] Q. Zou, L. Ni, T. Zhang, and Q. Wang. Deep learning based feature selection for remote sensing scene classification. *IEEE Geoscience and Remote Sensing Letters*, 12(11):2321–2325, 2015.