ENRICHING WORD REPRESENTATION LEARNING FOR AFFECT DETECTION AND AFFECT-AWARE RECOMMENDATIONS

NASTARAN BABANEJAD

A DISSERTATION SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

DECEMBER 2020

Abstract

The role of detecting affects from text is to detect affective states such as mood, sentiment and emotions from textual data like news articles or product reviews. The main affective tasks, including sentiment analysis, emotion classification and sarcasm detection have been popular in recent years due to a broad range of relevant applications in various domains such as healthcare, recommender systems, and hate speech detection. Traditionally, recommender systems deal with applications having only two types of entities, users and items, and do not put them into a context when providing recommendations. Recently, a family of recommendation algorithms has emerged to improve recommendations by adapting the contextual information of users and items. These models provide the promise of being more accurate as they are tailored to satisfy the continuously changing needs of users by considering more contextual information. However, little attention has been paid to the affective context and its relation to recommendations.

In this dissertation, we first investigate the impact of using affective information on the quality of recommendations, and then seek to improve affect detection in text

by enhancing word representation learning. We enrich word representations in two ways: one by effective pre-processing of the text corpora for training word embeddings and the second by incorporating both affective and contextual features deeply into text representations. We demonstrate the benefits of enriched word representations in both affect detection and affect-aware recommendation tasks.

This dissertation consists of five contributions. First, we investigate whether, how and to what extent emotion/sentiment features can improve recommendations. Towards that end, we derive a large number of emotion and sentiment features that can be attributed to both items and users in the domain of news and music. Then, we devise state-of-the-art emotion-aware recommendation models by systematically leveraging these features.

Second, we study the problem of pre-processing in word representation learning for affective tasks. Most early models of affective tasks employed pre-trained word embeddings. While pre-processing in affective systems is well-studied, text pre-processing for training word embeddings applied to affective systems, is not. To address this limitation, we conduct a comprehensive analysis of the role of text pre-processing techniques in word representation learning for affective analysis. Our investigation is the first of its kind and provides useful insights of the importance of each pre-processing technique when applied at the embedding training phase, commonly ignored in pre-trained word vector models, and/or at the downstream task phase.

Third, we investigate the usefulness of customized pre-processing for word represen-

tation learning for affective tasks. In particular, we study the role of each pre-processing factor in a specific affective task. We argue that using numerous text pre-processing techniques at once as a general combination for all affective tasks decrease the performance of affect detection. Therefore, we conduct extensive experiments, showing that, appropriate combination of text pre-processing methods for each affective task can significantly enhance the classifier's performance.

The fourth contribution of this dissertation seeks to dig deeper and study the role of affective and contextual embeddings with deep neural network models for affect detection. Early word embedding methods, such as Word2vec, are non-contextual, meaning that a word has the same embedding vector independent of its context and sense. Contextual embedding techniques, such as BERT (Bidirectional Encoder Representations from Transformers), solve this problem, but do not incorporate affect information in their word representations. We propose two novel deep neural network models that extend BERT to incorporate both affective and contextual features in text representations. We evaluate the two models in affect detection tasks and show the superior performance of the proposed methods for affect detection. Lastly, we show the usefulness of our proposed affective and contextual embedding models by applying them to affect-aware recommendations. In particular, we conduct a thorough experimental evaluation on real datasets in news and music domains to demonstrate the helpfulness of the proposed models in improving recommendations' performance.

Dedication

To My Wonderful Parents for Everything,

"To my mother "Nasrin ", in loving memory , who was my strongest pillar and constant source of inspiration."

&

"To my beloved father "Ahmad ", a strong and gentle soul who always taught me the best path to follow."

" In memory of Prof. Nick Cercone."

“ I might only have one match, but I can make an explosion "

- Rachel Platten (My Fight Song)

Acknowledgements

My PhD journey was made possible thanks to many wonderful people. Firstly, I would like to express my sincere gratitude to my advisor Prof. Aijun An for her dedicated and continuous support of my Ph.D study and related research, for her patience, motivation, and immense knowledge. I sincerely thank you for showing faith in me all those years, and for constantly inspiring, encouraging and supporting me over the years. I could not have imagined having a better advisor and mentor for my Ph.D study.

Besides my advisor, I would like to thank my co-advisor Prof. Manos Papagelis for his insightful comments and encouragement, and for his willingness and enthusiasm to assist in any way he could throughout the research project.

I would also like to extend my sincere thanks to my committee members, Prof. Natalija Vlajic and Prof. Jarek Gryz for their valuable advice. Your insightful comments not only purified my work, but also opened up an avenue for future research.

I thank my labmates for the interesting discussions and the sleepless nights we were working together before deadlines. In particular, I am grateful to Dr. Heidar Davoudi and

Dr. Ameeta Agrawal for always supporting me in any ways they could. I am also thankful to the administrative and technical staff of our department at York University.

I would like to thank The Globe and Mail for providing the datasets used in this work. This work is funded by Natural Sciences and Engineering Research Council of Canada (NSERC), The Globe and Mail, and the Big Data Research, Analytics and Information Network (BRAIN) Alliance established by the Ontario Research Fund - Research Excellence Program (ORF-RE).

And finally, the endless love and precious support of my awesome family and best friend cannot be appreciated enough in words. To my late mother, who could not witness my success. Although she is not here to give me strength and support, I always feel her presence that used to urge me to strive to achieve my goals in life – she inspired me everyday and taught me how to be strong and face the challenges with faith and humility, as she used to say – *" where there is a will there's a way ! "*. To my father, who always had confidence in me, encouraged me to chase my dreams and supported me in all my endeavors - now *I promise not to staying up all night again!*. To my sisters Ghazaleh and Ghoncheh, who offered invaluable support and humor over the years. Profound thanks to you all to putting up with my stresses and moans through those years – I would not have been where I am today without you. Last but by no means least, to my partner and best friend Majid Taghdimi, who put up with my idiosyncrasies and all the ups and downs for the past three years! – I am grateful for all your help and support, *Hakuna Matata!*

Table of Contents

Abstract ii

Dedication v

Acknowledgements vii

Table of Contents ix

List of Tables xviii

List of Figures xxiii

1 Introduction 1

1.1 Background and Motivation 1

1.1.1 Affect Detection in Text 3

1.1.2 Affect-Aware Recommendation Model 9

1.2 Research Problems and Scope 12

1.3 Research Contributions 14
1.3.1 Leveraging Emotion Features in Content Recommendations . . 14
1.3.2 A Comprehensive Analysis of Pre-processing for Word Representation Learning in Affective Tasks 15
1.3.3 Customized Pre-processing for Word Representation Learning in Affective Tasks 16
1.3.4 Affective and Contextual Embedding for Affect Detection . . . 17
1.3.5 Affect-aware Recommendation 18
1.3.6 The Publications 20
1.4 Dissertation Organization 20

2 Literature Review **22**

2.1 Leveraging Emotion Features in Recommender Systems 22
2.1.1 Emotion in News Recommender Systems 23
2.1.2 Emotion in Music Recommender Systems 24
2.2 Pre-processing for Word Representation Learning in Affective Tasks . . 25
2.2.1 Pre-processing Classification Datasets 26
2.2.2 Pre-processing Word Embeddings 27
2.3 Affective and Contextual Embedding for Affect Detection 29
2.3.1 Affective Features 29

2.3.2 Contextual Features 30
2.3.3 Affective-Contextual Features 32
2.3.4 Task-specific Corpora for Sarcasm Detection 33

3 Leveraging Emotion Features in Social Media Recommendations 35

3.1 Introduction 35
3.2 Features for recommendation 41
3.2.1 Emotion-based Features 41
3.2.1.1 Item Emotion-based Features 42
3.2.1.2 User Emotion-based Features 45
3.2.2 Non-Emotion-based Features 46
3.2.2.1 Item Non-Emotion-based Features 46
3.2.2.2 User Non-emotion-based Features 47
3.2.3 Feature Selection 49
3.3 Emotion-aware Recommendation Model (EmoRec) 50
3.3.1 Model Training 50
3.3.1.1 Model 1 (Boost Model) 51
3.3.1.2 Model 2 (Deep Neural Network (DNN)) 52
3.3.1.3 Model 3 (Deep Matrix Factorization (Deep MF)) . . . 53
3.4 Experimental Evaluation 55

3.4.1 Data 55
3.4.2 Evaluation Metrics 56
3.5 Discussion and Analysis 57
3.5.1 Comparing Recommendation Models with and without Emotion Features 57
3.5.2 Comparison with Other Baselines 60
3.5.3 Effect of Individual Emotion Features 63
3.6 Summary 65

4 A Comprehensive Analysis of Pre-processing for Word Representation Learning in Affective Tasks 67

4.1 Introduction 67
4.2 Pre-processing in Affective Systems 71
4.2.1 Pre-processing Factors 71
4.2.2 Order of Pre-processing Factors 73
4.3 Experimental Evaluation 74
4.3.1 Training Corpora 74
4.3.2 Word Embedding Models 76
4.3.3 Evaluation Datasets 78
4.3.4 Classification Setup 81

4.4 Discussion and Analysis 82
4.4.1 Effect of Pre-processing Factors 82
4.4.2 Evaluating Pre-processing Training Corpora vs. Pre-processing Classification Dataset 86
4.4.3 Evaluating Proposed Model against State-of-the-art Baselines . 89
4.4.4 Analyzing the Three Affective Tasks 91
4.5 Summary 92

5 Customized Pre-processing for Word Representation Learning in Affective Tasks 93

5.1 Introduction 93
5.2 Customized Pre-processing in Affective Tasks 97
5.2.1 Sentiment Analysis 98
5.2.2 Sarcasm Detection 100
5.2.3 Emotion Detection 101
5.2.4 Pre-processing Factors 101
5.3 Customized Pre-processing Training Corpora 104
5.4 Customized Pre-processing Classification Tasks 104
5.5 Experimental Evaluation 105
5.5.1 Training Corpora and Evaluation Datasets 105

5.5.2 Word Embedding Models 105
5.5.3 Classification Setup 106
5.6 Discussion and Analysis 107
5.6.1 Effects of General Combination of Pre-processing Factors . . . 108
5.6.2 Analyzing the Three Affective Tasks 113
5.6.3 Effects of Customized Pre-processing Factors for Each Affective Task 114
5.6.4 Evaluating Pre-processing Training Corpora vs. Pre-processing Classification Dataset 116
5.7 Summary 120

6 Affective and Contextual Embedding for Affect Detection 121

6.1 Introduction 121
6.2 Proposed Models for Sarcasm Detection 126
6.2.1 Affective Feature Embedding (AFE) 127
6.2.1.1 Affective Feature Vector Representation 128
6.2.1.2 Bi-LSTM Layer 130
6.2.1.3 Multi-head Attention Layer 131
6.2.2 Contextual Feature Embedding in ACE 1 132
6.2.2.1 Training BERT 132

6.2.2.2 Training Affective BERT with Affective Feature Embeddings . . . 133
6.2.2.3 Fine-tuning BERT Models . . . 135
6.2.3 Contextual Feature Embedding in ACE 2 . . . 135
6.2.3.1 Pre-trained Embeddings . . . 136
6.2.3.2 Obtaining Sentence Embeddings Using SBERT . . . 136
6.2.3.3 Combining the Two Components . . . 137
6.3 Experimental Evaluation . . . 138
6.3.1 Corpora for Training Embeddings . . . 138
6.3.2 Affective Tasks Datasets . . . 139
6.3.3 Experimental Setup . . . 140
6.4 Discussion and Analysis . . . 142
6.4.1 Comparing Variations of ACE 1 and ACE 2 . . . 142
6.4.2 Evaluating Proposed Models against State-of-the-art Baselines . 146
6.5 Evaluating the Performance of Proposed Models on Other Affective Tasks 156
6.6 Summary . . . 160

7 Affective and Contextual Embedding Model for Feature Representation Learning in Affect-Aware Recommendation 162

7.1 Introduction . . . 162

7.2 Recommendation Algorithms 164
7.3 Affective and Contextual Feature Representation in Recommendation Models 167
7.3.1 Affect-Aware Recommendation Model (AARec) 168
7.4 Experimental Evaluation 172
7.4.1 Datasets 173
7.4.2 State-of-the-art Recommendation Algorithms 173
7.5 Discussion and Analysis 174
7.5.1 Evaluating the Effects of Proposed Affect Detection Methods in Recommendation Algorithms 174
7.5.2 Evaluating the Performance of AARec Against EMoRec 177
7.6 Summary 178

8 Conclusions and Future Directions **180**

8.1 Summary of Approaches and Contributions 181
8.2 Future Directions 186
8.2.1 Recommendation with Affective Information Through Other Cues 186
8.2.2 Negation Scope and Negation Handling 187
8.2.3 Multilingual Model 188
8.2.4 Learning of Affective Representations Through Graphs 188

8.2.5 Integrating the Proposed Models into One System 189

Bibliography **190**

List of Tables

3.1 Emotion Resources 43

3.2 List of Emotion Feature Importance 48

3.3 List of Non-emotion Feature Importance 49

3.4 Results of our Models on News Dataset (F-score) 59

3.5 Results of our Models on Music Dataset (F-score) 60

3.6 Comparison of EMOREC with State-of-the-art Baselines on News Dataset (F-score) 62

3.7 Comparison of EMOREC with State-of-the-art Baselines on Music Dataset (F-score) 62

3.8 Effect of Individual Emotion Features (F-score) 64

3.9 Effect of Top Three Emotion Features (*Plutchik emotions, User emotions across categories, and User emotions across items*) on State-of-the-art Models 65

4.1 Details of News training corpora 75

4.2 Details of Wikipedia training corpora 76
4.3 Details of evaluation datasets 77
4.4 Examples of text instances in the evaluation datasets 79
4.5 F-score results of evaluating the effect of pre-processing factors using CBOW on News corpus. The overall best results are in **bold**. The best result using only any one pre-processing setting is underlined. 83
4.6 F-score results of evaluating the effect of pre-processing factors using Skip-gram on News corpus. The overall best results are in **bold**. The best result using only any one pre-processing setting is underlined. 84
4.7 F-score results of evaluating the effect of pre-processing factors using CBOW on Wikipedia corpus. The overall best results are shown in **bold**. 85
4.8 F-score results of evaluating the effect of pre-processing factors using Skip-gram on Wikipedia corpus. The overall best results are shown in **bold**. 86
4.9 F-score results of evaluating the effect of pre-processing factors using BERT on Wikipedia corpus. The overall best results are shown in **bold**. 87
4.10 F-score results of evaluating the effect of pre-processing word embeddings training corpus vs. pre-processing evaluation datasets 88
4.11 F-score results of comparing against state-of-the-art word embeddings. The best score is highlighted in **bold**, and the second best result is underlined. 89

5.1 Example of Case Studies of Different Pre-processing in Affective Tasks. 99

5.2 F-score Results of Evaluating the Effect of Different Pre-processing Factors Using Different Models on Wikipedia Corpus. The Overall Best Results Are in **Bold**. 108

5.3 F-score Results of Evaluating the Effect of Different Pre-processing Factors Using Different Models on Wikipedia Corpus. The Overall Best Results Are in **Bold**. 110

5.4 F-score Results of Evaluating the Effect of Pre-processing Word Embeddings Training Corpus vs. Pre-processing Evaluation Datasets 112

5.5 F-score Results of Comparing the Effect of Customized Pre-processing Vs General Pre-processing 115

5.6 F-score Results of Evaluating the Effect of Customized Pre-processing Word Embeddings Training Corpus vs. Customized Pre-processing Evaluation Datasets 118

5.7 F-score Results of Evaluating the Effect of Customized Pre-processing Word Embeddings Training Corpus vs. Customized Pre-processing Evaluation Datasets 119

6.1 Description of sarcasm detection datasets. 140

6.2 F1-scores of model ACE 1 with different settings. Best results are in **bold** and 2nd best are underlined. 143

6.3 F1-scores of model ACE 2 with different settings. Best results are in **bold** and 2nd best are underlined. 145

6.4 F1-score results of comparing different pre-trained embeddings with different affective embeddings for each model. The best score is highlighted in **bold**, and the second best result is underlined. 146

6.5 F1-scores for comparing our models against state-of-the-art models (Only Affective). The best scores are in **bold**, and 2nd best are underlined, while the 3rd best are double underlined. 147

6.6 F1-scores for comparing our models against state-of-the-art models (Only Contextual with Fine-Tune). The best scores are in **bold**, and 2nd best are underlined, while the 3rd best are double underlined. 148

6.7 F1-scores for comparing our models against state-of-the-art models (Only Contextual with Pre-trained without Fine-Tune). The best scores are in **bold**, and 2nd best are underlined, while the 3rd best are double underlined. 149

6.8 F1-scores for comparing our models against state-of-the-art models (Affective-Contextual). The best scores are in **bold**, and 2nd best are underlined, while the 3rd best are double underlined. 153

6.9 F1-scores of model ACE 1 with different settings on other affective tasks. Best results are in **bold** and 2nd best are underlined. 158

6.10 F1-scores of model ACE 2 with different settings on other affective tasks. Best results are in **bold** and 2nd best are underlined. 159

6.11 F-score Results of comparing ACE 1 and ACE 2 against the customized pre-processing model. 160

7.1 Comparison of The performance of Different Recommendation Models Using Different Modes on Music Dataset (F-score) 176

7.2 Comparison of The performance of Different Recommendation Models Using Different Modes on News Dataset (F-score) 176

7.3 F-score Comparison of EMOREC Performance Against AAREC on News and Music Dataset 178

List of Figures

3.1 Illustrative example of emotions expressed in articles read by two different users, U_1 (left) and U_2 (right), over a three month period. Can we leverage the emotional context to improve recommendations? 36

3.2 Overview of an emotion-aware recommendation system. 39

3.3 Example emotions expressed in textual content 42

3.4 The Structure of Our DNN Model 52

3.5 The Structure of Our Deep MF Model 54

4.1 Framework of applying pre-processing in different stages in affective systems; (a) Pre, (b) Post. 69

4.2 Absolute F-scores vs. relative improvement 91

5.1 Average F-scores vs. relative improvement 113

6.1 Overview of the proposed model ACE 1 126

6.2 Overview of the proposed model ACE 2 127

6.3 Overview of the proposed model ACE 2 129

6.4 BERT Input Representation Vs Affective BERT Input Representation. . 134

7.1 Different Recommendations Based on The Recommendation Model Algorithm. 165

7.2 The Architecture of the Sequential Hybrid Attention-Based Model (SHAN) Proposed in [200] 170

1 Introduction

1.1 Background and Motivation

In recent years, there has been increasing interest in affect analysis from both academia and industry. In general, affect analysis involves the related tasks of sentiment analysis, emotion detection and sarcasm detection, amongst others. This surge of activity is due to the rapid growth of social networks, product reviews, news media, blogs, forum posts, and the consequent easy access to a mass of digitally documented subjective and emotional data. Therefore, a vast amount of affective information can be easily obtained from forums, blogs, review sites and other websites. However, as helpful as these user-generated contents are, it is not a straightforward job to find, evaluate, and interpret all these data manually to get to the heart of the matter. Moreover, emotions are widely recognized as key factors in decision-making process based on cognitive psychology [68]. Therefore, automatic affect analysis can lend a hand to improve decision-making systems such as news recommendation systems.

In addition, according to early literature, affect has been recognized as an essen-

tial factor that influences users' behaviour [23, 146]. With affective information about users/items, recommendation systems will be able to recommend more appropriate items that match users' needs [33, 70]. Thus, it would be beneficial to incorporate users' contextual and affective information in the recommendation process. However, detecting affects in the text is a challenging task, which is one of the primary reasons for a comparatively limited number of previous studies on the use of affective information in recommender systems. Developing a strong relationship between users/items affective context and recommendation models is not a trivial task in the competitive world of digital media. Therefore, recommendation models need to build effective strategies to identify, extract and select the most relevant affect-based features for use in recommendation models. In order to achieve the aforementioned objectives, recommender systems should gain insight into users' needs and behaviour as well as items based on affective detection methods. This is where natural language processing techniques come to play. In this dissertation, we focus on affect extraction and detection methods and the use of extracted affect information in recommendation models. In particular, our goal is first to investigate the impact of using manual extraction of affective information on the quality of recommendations and then seek to improve affect detection from the text by enhancing word representation learning. To that end, we frame the challenges into different research problems and propose effective solutions for them accordingly.

1.1.1 Affect Detection in Text

In natural language processing, the analysis of affect, feelings, emotions, sentiments, and opinions includes the tasks of sentiment analysis (usually concerning the binary or tertiary classification of text into categories of positive and negative sentiments, and sometimes neutral as well), emotion detection (which entails classifying text into one or more categories of emotion such as happiness, sadness, anger, etc.) and sarcasm detection (in which text is classified as being sarcastic or not) [166].

There has been a growing interest in developing computational methods for affect detection from a text in recent years. This interest can be largely attributed to the fundamental and straightforward nature of the methods employed, the availability of vast amounts of user-generated natural language data, and the wide range of useful applications, spanning from hate speech detection [41] to monitoring the sentiment of financial markets [67, 178] and recommendation models [13, 91]. While much study has been done in the field, this task remains far from being solved. The difficulty is mainly due to the fact that the occurrence of affect is only in a very small portion of cases marked by the presence of affect-related words.

Word embedding is an effective word representation method that reflects aspects of word meaning and helps improve the accuracy of the classification in various NLP tasks [1, 116]. This technique is to study the continuous representations of words in a

low-dimensional vector space by leveraging the contextual information from large corpora [116]. Some recent affect analysis models employed pre-trained word embeddings obtained under the assumption of the distributional hypothesis in word representation learning models [45, 115]. The distributional hypothesis suggests that two words that often occur in similar linguistic contexts appear to be more semantically similar and should be interpreted in the embedding space closer to each other. However, while such embedding is useful for many downstream tasks of natural language processing (NLP), it is considered less suitable, particularly for affective tasks [4, 172]. For example, `word2vec` [115] estimates the pair of words 'happy' and 'sad' to be more similar than the pair of words 'happy' and 'joy', which is counter-intuitive and might affect the accuracy performance of the models that depend on it. The main side effect of this issue is that the terms with opposite sentiment polarity are transformed into near vectors, and it is a challenging issue in affective tasks such as sentiment analysis and emotion detection. Several techniques have been proposed to address the limitations of traditional word embeddings, including task-specific fine-tuning [45], retrofitting [52], and generating affective word embeddings [53], as well as fine-tuned pre-processing strategies tailored to different NLP tasks, to name a few. While these strategies have demonstrated evidence of improving the accuracy performance in tasks such as word similarity, word analogy, and others [106], their effect in affective tasks has not received considerable attention and remains less explored due to numerous reasons. One of the most common challenges of affect detection is negation,

where in linguistics, negation is a way of reversing the polarity of words, phrases, and even sentences [211]. For instance, negating words such as *"no"*, *"not"*, *"shouldn't"* are very important linguistics because they affect the polarities of other words.

The advantages of emotion analysis in political science [48], psychology, marketing [16], human-computer interaction [23], and recommender systems [210], gave the field of emotion detection in NLP life of its own, resulting in a surge of many studies in recent years. A common approach to emotion analysis and modeling is categorization, e.g., according to Ekman's basic emotions, namely, anger, disgust, fear, happiness, sadness, and surprise [51]. In particular, emotion analysis is further complicated than binary or tertiary classifications such as sentiment analysis due to the greater number of categories (emotions) involved to undertake classification [3]. Categorization into particular emotion classes is more troublesome not just on the grounds that emotion detection generally requires in-depth insights but also because of the similar nature of different emotions such as *anger* and *disgust*, which makes the classification a challenge even for human annotators [9].

Furthermore, emotions from a text can be perceived at different levels, including a word, a phrase, a sentence, a paragraph, or even an entire document. In general, proposed models to detect emotions from any level of text (e.g. sentence, document) are instantiated with word-level analysis to some extent and then expanded to larger units of text.

Early studies in emotion detection used keyword-based and lexicon-based approaches

for detecting emotions in text. However, these techniques have limitations, e.g., the absence of a particular emotion-bearing keyword in text, as well as a single word that can elicit multiple emotions depending on the context in which they are used. Consider an example sentence, *"He kills dogs"*, consisting of an emotion-evoking word, i.e., *"kills"*, mostly associated with the emotions of *fear* and *anger*. Although a word may evoke different emotions, it can only be disambiguated in context (e.g., *"Your smile kills me"*).

Recently, machine learning methods, especially deep neural networks, have been proposed to overcome the limitations of previous approaches, mainly by adopting word embedding vectors that represent rich semantic/syntactic information for many affective tasks. However, as it was mentioned before, those models fail to capture the emotional similarity information when they are used directly in the classification tasks [4]. Thus, pre-processing techniques such as punctuation and negation [155] or pos-tagging and negation [160] make up a common component of many emotion classification models [95, 140]. Moreover, prior studies also indicated that interjections, which is common in textual data, can be detected for potential emotions [61, 88, 188].

Hence, the successful achievements of various text pre-processing in affective tasks raise interesting questions: *which pre-processing techniques yield the most benefit in affective tasks?* and *what is the effect of integrating pre-processing techniques earlier into word embedding models instead of post-processing classification tasks?* Despite the importance of this question and potential benefits of pre-processing techniques in affective

tasks, there is no comprehensive research on using such techniques in different stages of word representation learning or downstream tasks on classification datasets. We consider these components and conduct a comprehensive analysis of the role of pre-processing techniques in affective tasks (including sentiment analysis, emotion classification and sarcasm detection), employing different word representation learning models. Moreover, we perform a comparative analysis of applying these techniques in different stages of word embeddings rather than downstream tasks while studying the benefits of different combinations of customized pre-processing for each affective task individually.

Sarcasm detection on social media has attracted much interest in the community of natural language processing over the past decade with its own challenges. Sarcasm is the particular form of language that happens when someone conveys implicit information, usually having the opposite meaning of what is said or written [163]. Because of this deliberate ambiguity, it is a particularly difficult task to detect sarcasm, especially in written interactions where usual hints of sarcasm such as body gestures, the tone of voice or facial expression are not present [86, 163].

Early attempts for sarcasm detection from text mainly relied on looking for a set of positive verbs and negative/undesirable situations (e.g. “I love [positive verb] the pain of breakup [negative situation]”) [63, 154] or using lexical features such as n-grams, capital letters, excessive usage of exclamatory marks, usage of emoticons, punctuation or syntactic and pattern-based features [108], to name a few. Later attempts for sarcasm

detection mostly relied on language models that are based on continuous representation or embeddings of words, such as Word2vec [116] and GloVe [142]. The use of these general models can eliminate the need for lots of pre-processing, and manual feature engineering or dependence on enormous emotion labelled datasets. However, due to the mechanism with which word vectors are learned and embedded into a space, these models have been shown to be inadequate for affective tasks [10]. Contextual embedding techniques, such as BERT (Bidirectional Encoder Representations from Transformers), solve this problem but do not incorporate affect information in their word representations. As such, an interesting question is that *can we automate sarcasm detection task using a deep neural network model by incorporating both affective and contextual features of text?* While this question is crucial to any sarcasm detection using deep neural networks, to date, it has not been investigated if incorporating affective representation with contextual learning models such as BERT at the training phase to train the model from the ground-up will improve the performance of the sarcasm detection task. An answer to this question can provide benefits for many applications, not only for the task of sarcasm detection but also to other affective tasks. For example, it can be used to identify different types of affect, emotions and sentiments in documents that can be later used for any recommendation models. However, predicting sarcasm in the text needs understanding the interplay between the context and affective representation of a document. We consider these elements and propose a deep neural network architecture by incorporating both affective and contextual

features of text to build a classifier that can determine whether a document is sarcastic or not. Furthermore, we investigate if the proposed model can improve the performance of other affective tasks, such as emotion detection and sentiment analysis.

1.1.2 Affect-Aware Recommendation Model

Recommendation systems have become ubiquitous over the last decade, providing users with personalized recommendations on video streams, news excerpts, and purchasing hints. In the pursuit of increasing the performance of recommendation systems, researchers started to turn to more customer-driven and context descriptors in recent years. The advances made in affect detection, especially in automatic emotion detection and sentiment analysis techniques, paved the way for the utilization of affects and personality as descriptors that account for a larger part of the variance in user preferences than the generic descriptors (e.g. history) used so far [82, 89]. Human emotions are generally considered the primary predictors of actions and preference [102, 103]. They are a key factor in decision-making [68], but very little has been learned until recently about the usefulness of using affect information in customizing real-world recommendation systems [120, 175]. However, these research efforts have been conducted independently, stretched among the two major research areas, recommender systems and affective analysis. Furthermore, affects are also very difficult to identify, quantify and measure precisely, which is one of the primary reasons for a relatively small number of previous work on using

affective information directly in recommendation systems. However, it can be beneficial for a recommender system application to detect and make fair use of affective information.

The emotional profile of a user can be determined through explicit or implicit feedback of users to items. Explicit feedback, such as providing ratings and/or submitting reviews on items, can represent an accurate reflection of a user's opinion about the item. However, it is considered as an intrusive process that disrupts the user-system interaction and negatively impacts user experience [135]. Moreover, while it might be available for certain domains (e.g. product recommendations [34], movie recommendations [131], etc.), it is not easily obtainable in domains such as news, where users typically interact with items at a fast pace and are less inclined to provide feedback. In the context of recommendation systems, through extracting and combining the term frequency and inverse document frequency (tf*idf) from song lyrics, researchers [36] constructed a composite emotion point matrix for each song, which is then used to further classify songs based on their emotion and make a recommendation accordingly. In [191], an emotion-aware recommendation approach is proposed which analyzes emotion based on the song lyrics and user comments via vector projection. More recently, Mizgajski et al. [120] introduced a recommender system for recommending news items by leveraging a multi-dimensional model of emotions, where emotion is derived through user's self-assessed reactions (i.e., explicit feedback), which can be considered as an intrusive collection. When the explicit feedback is absent, sparse, or costly to obtain, incorporating implicit feedback, which is generally abundant and

non-intrusive, might be beneficial.

Inspired by these observations, recent advancement in methods for affect detection and the success of affective feature extraction in recommendation algorithms, the interesting questions are *whether*, *how* and *to what extent* affect features can improve the accuracy of recommendations. We consider these elements and use various affect detection approaches to identify and extract the most relevant affect-based features for recommendation models.

Furthermore, previous works showed different affect detection approaches that can help to build an emotional profile of users and content items have different effects on the performance of content-based recommender algorithms [13, 80]. In [90, 118], authors showed successful identification of sarcastic utterance in the users' reviews can enhance the personalization of content ranking and recommendation systems. At the same time, others [176] indicated that the automatic detection of emotions and sentiments in a user's review could be useful as an indicator of the user's satisfaction with the content.

In addition, the quality of the affective features in the data directly affects the predictive model and the results it can be achieved [98, 209]. As such, the first interesting question is: *can we benefit from utilizing the more successful affect detection methods in the modeling process to make the most relevant recommendations to the users?* An answer to this question can provide benefits not only to affect detection itself but also for many other applications such as in the news and music recommendations. In fact, answering this question is challenging since it needs more appropriate techniques to identify and

extract these affective features. Thus, it raises other questions as: *which affect detection techniques are more effective in identifying affects in text documents?* and *after identifying the potential affective features, how do we incorporate these affective information in recommendation models to see which affect detection approaches yield the most benefit in recommendation systems?* While these questions are crucial to any recommendation model using the affective features, to date, it has not been investigated from the natural language processing perspective. We consider these findings and propose a deep neural network architecture to utilize them for affect detection. Moreover, we exploit the proposed approaches in affect detection, which can serve as the criteria in the recommendation model.

1.2 Research Problems and Scope

Affect detection from text and recommendation systems are quite challenging, and complicated issues. The term of affect detection is broad in natural language processing as it can be studied from different discipline perspectives (e.g., emotion detection, sentiment analysis, sarcasm detection). In this research, we consider all three aspects of affect detection and attempt to improve their performance on real-world datasets. Furthermore, we utilize the proposed approaches for affect detection in the recommendation models to demonstrate the usefulness and benefits of affect detection in news and music recommendations.

To that end, we investigate the following open research questions, and design cutting-edge data-driven approaches to solve these problems:

- **Problem 1:** Can the use of affective information in text improve the performance of content (e.g., news or music) recommendation systems? If yes, how and to what extent can affective features improve the accuracy of recommendations?

- **Problem 2:** What is the effect of integrating text pre-processing techniques earlier into word embedding models, instead of later on in downstream classification models, on the accuracy of affect detection? Which pre-processing techniques yield the most benefit in affective tasks?

- **Problem 3:** Will incorporating both affective and contextual features deeply into text representations using a deep neural network architecture improves the performance of affect detection?

- **Problem 4:** Can improving the affect detection approaches in text and enriching word representation learning improve the performance of affect-aware recommendations?

Our overreaching goal in this dissertation is to demonstrate the effects of affective information in recommendation models by improving the affect detection techniques in text and enhancing word representation learning, which is essential for any affect

analysis approaches using deep neural networks. We enrich word representations in two ways: one by effective pre-processing of the text corpora for training word embeddings and the second by incorporating both affective and contextual features deeply into text representations.

1.3 Research Contributions

The contributions of this dissertation fall into different categories, which are summarized in the subsequent sections.

1.3.1 Leveraging Emotion Features in Content Recommendations

In order to evaluate the importance of the emotional context to recommendations, we have to incorporate various emotion/sentiment features to recommendation algorithms and evaluate their accuracy performance. The major contributions in this line of research are as follows:

- We systematically identify, extract and select the most relevant emotion-based features for use in recommendation models. These features are associated with both items and users in the domain.

- We devise a number of state-of-the-art models for generating recommendations that incorporate the additional emotion features. In addition, we use ensembling

methods to increase the predictive performance by blending or combining the predictions of multiple constituent models.

- We propose EMOREC, an emotion-aware recommendation model, which demonstrates the best accuracy performance in news and music recommendation task. EMOREC itself is an ensemble model.

- We conduct a thorough experimental evaluation on real datasets coming from diverse domains (news and music). Our results demonstrate that the emotion-aware recommendation models consistently outperform state-of-the-art non-emotion-based recommendation models.

1.3.2 A Comprehensive Analysis of Pre-processing for Word Representation Learning in Affective Tasks

The overarching goal of this research is to perform an extensive and systematic assessment of the effect of a range of linguistic pre-processing factors on three affective tasks, including *sentiment analysis*, *emotion classification* and *sarcasm detection*. Towards that end, we systematically analyze the effectiveness of applying pre-processing to large training corpora *before* learning word embeddings, an approach that has largely been overlooked by the community. The main contributions of this work are as follows:

- We conduct a comprehensive analysis of the role of pre-processing techniques in

affective tasks (including sentiment analysis, emotion classification and sarcasm detection), employing three word embedding models, over nine datasets;

- We perform a comparative analysis of the accuracy performance of word vector models when pre-processing is applied at the training phase (training data) and/or at the downstream task phase (classification dataset). Interestingly, we obtain the best results when pre-processing is applied only to the training corpus or when it is applied to both the training corpus and the classification dataset of interest.

- We evaluate the performance of our best pre-processed word vector model against state-of-the-art pre-trained word embedding models.

1.3.3 Customized Pre-processing for Word Representation Learning in Affective Tasks

We propose a model using various combinations of different pre-processing methods for each affective task individually using different word embedding models. Major contributions of this work are as follows:

- We investigate the usefulness of customized pre-processing for word representation learning in affective tasks. In particular, we propose different combinations of pre-processing factors which are most suitable for each affective task, employing seven embedding models, over nine datasets.

- We study the role of each combination of pre-processing factors for a specific affective task and their major effects on the performance when they are applied in different stages of word embedding.

- We conduct extensive experiments, showing that the appropriate combination of text pre-processing methods for each affective task when they are applied in different stages of word embedding can significantly enhance the classifier's performance.

- We perform a comparative study comparing the accuracy performance of word vector models proposed in this work and that of the one presented in the previous model.

1.3.4 Affective and Contextual Embedding for Affect Detection

We propose two novel models that incorporate contextual and affective features in a deep neural network architecture for affect detection. The main contributions of this work are as follows:

- We present two novel deep neural network language models (ACE 1 and ACE 2) for affect detection. Each model extends the architecture of BERT by incorporating both affective and contextual features of text to build a classifier to detect affects in a document.

- Integral to our proposed models is a novel model that learns the affective representation of a document, using a Bi-LSTM architecture with multi-head attention. The resulting representation takes into account the importance of the affect representations of the sentences in the document.

- We design and evaluate alternatives that materialize each of the two components (affective feature embedding and contextual feature embedding) of the proposed deep neural network architecture model. We systematically evaluate the effectiveness of each alternative architecture.

- We conduct an extensive evaluation of the performance of the proposed models (ACE 1 and ACE 2), which demonstrates that they significantly outperform current state-of-the-art models for the affective task of sarcasm detection.

- We investigate whether the proposed affective and contextual model can improve the performance of other tasks, such as emotion detection and sentiment analysis. Furthermore, we form a comparative study of the accuracy performance of previous model using pre-processing with the current proposed model.

1.3.5 Affect-aware Recommendation

We evaluate the performance and usefulness of our proposed affective and contextual embedding models by applying them on recommendation algorithms and propose an

affect-aware recommendation using the attention mechanism. The proposed model outperforms the state-of-the-art models in the prediction task. The main contributions are as follows:

- We design an affect-aware recommendation model by adopting a Sequential Hierarchical Attention Network (SHAN) for recommending the next items. In particular, we combine user's long and short-term preferences to generate a high-level hybrid representation of users' and items' affective information using our proposed models ACE 1 and ACE 2.

- We present a comparative study measuring the performance of EMoRec proposed earlier against the affect-aware recommendation (AARec).

- The extensive experiments are conducted to show the effectiveness of the proposed model of affect analysis against the state-of-the-art baseline approaches in recommendation systems.

All the proposed models for recommendations are evaluated on real datasets from The Globe and Mail, a major newspaper in Canada, and KKBOX, which is Asia's leading music streaming service provider. However, the proposed techniques are not limited to these two datasets and can be used with other news or music lyrics data. Therefore, the models are general and can be applied by other applications to improve the accuracy of the recommendations.

1.3.6 The Publications

The outcomes of this study have been published in the proceedings of the top tier conferences in the field of natural language processing such as the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020) [12], the 28th International Conference on Computational Linguistics (COLING 2020) [14], and the 7th International Workshop on News Recommendation and Analytics (INRA 2019) in conjunction with 13th ACM Conference on Recommender Systems (RecSys 2019) [13].

1.4 Dissertation Organization

This dissertation is organized as follows. In chapter 2, we categorize and discuss the literature related to the different proposed models. Affect detection serves as an important component in our framework since our proposed models are built on them. As such, we introduce the concepts of different types of affect analysis and discuss their potential in recommendation models by proposing an Emotion-aware Recommendation model (EmoRec) in chapter 3. In chapter 4, we present a comprehensive analysis of pre-processing for word representation learning in affective tasks. This model aims to indicate the role of pre-processing methods when they are applied in different stages of word representation learning models. The goal of the customized pre-processing model in affective tasks is to demonstrate the importance of specific combinations of pre-processing

techniques which are required for each affective task. We explain this model in detail with related case studies in chapter 5. Chapter 6 discusses the proposed affective and contextual embedding model for affect detection. The usefulness of the proposed affect analysis models in an affect-aware recommendation is discussed in Chapter 7. Finally, in Chapter 8, we review the contributions, conclude the dissertation, and summarize the future research directions.

2 Literature Review

This dissertation designs a set of novel natural language processing/machine learning techniques as the algorithmic tools to help music streams and digital newspapers recommendation systems. A variety of areas in machine learning and natural language processing ranging from sentiment analysis, emotion detection, sarcasm detection to word representation learning with neural networks and deep learning have been considered in designing the proposed approaches. In order to facilitate studying the related work, we categorize it based on the main components of the dissertation. We study emotion features in recommender systems used in the literature in §2.1. The pre-processing in word representation learning for affective tasks are presented in §2.2. We discuss the related work in affective and contextual sarcasm detection models in §2.3.

2.1 Leveraging Emotion Features in Recommender Systems

According to cognitive psychology, emotion has been recognized as one of the important elements of human nature that has a significant impact on our behaviour and choices

[68, 103]. A number of studies in the area of psychology, neurology, and behavioural sciences have shown that individuals' choices are related to their feelings and mental moods [68, 122]. Gonzalez et al. [62] was one of the earliest works to use emotional context in recommender systems. They pointed out that emotions are crucial for users' decision making and that users transmit their decisions together with emotions. Below, we discuss the role of emotions in recommender systems in two specific domains of interest: news and music.

2.1.1 Emotion in News Recommender Systems

Prior research has found a range of features to be useful in the context of news recommender systems, such as user location [57], time of the day [125], demographic information [100], or article social media profile [212]. However, based on cognitive psychology, emotion, which is one of the essential elements of human nature that significantly impacts our behaviour and choices [103], has received little attention in recommendations. A number of studies in the area of psychology, neurology, and behavioural sciences have shown that individuals' choices are related to their feelings and mental moods [122].

In the context of recommender systems, one of the earliest works, [62], pointed out that emotions are crucial for users' decision making and that users transmit their decisions together with emotions. [176] introduced a unifying framework for using emotions in user interactions with a recommender system, and suggested that while an implicit approach

of user feedback may be less accurate, it is well suited for user interaction purposes since the user is not aware of it [176].

While emotions as features have been studied in movie recommendations [127, 131], music recommendations [71] and restaurant recommendations [181], to name a few, much less work has explored the role of emotion features in news recommender systems.

Emotion in news articles has been studied for categorizing news stories into eight emotion categories [7]. Specifically for recommender systems, [138] introduced a model for Persian news utilizing both the emotion of news as well as user's preference. More recently, [120] introduced a recommender system for recommending news items by leveraging a multi-dimensional model of emotions, where emotion is derived through user's self-assessed reactions (i.e., explicit feedback), which can be considered as an intrusive collection. In contrast to previous studies, our work focuses on studying the role of emotion features in news recommender systems using implicit user feedback.

2.1.2 Emotion in Music Recommender Systems

Chen and Tang [36] present a classification model for Chinese song recommender system based on computational analysis of the lingual part of song lyrics. Through extracting and combining the term frequency and inverse document frequency (tf*idf) from song lyrics, they construct a composite emotion point matrix for each song, which is then used to further classify songs based on their emotion and make a recommendation accordingly.

Hu et al. [81] proposed a method for detecting the emotions of Chinese song lyrics based on an affective lexicon and fuzzy clustering. Xie and Tang [191] proposed an emotion-aware recommendation approach that analyzes emotion based on the song lyrics and user comments via vector projection. The above-mentioned approaches have been designed for and evaluated on Chinese text data, which cannot be ported to English data in a straightforward manner. For English music, Gossi and Gunes [64] consider a recommendation system based on user-generated tag data representing genre, mood and gender of vocalist. Zangerle et al. [201] leveraged hashtags indicating sentiment provided by users for music recommendation. As user-generated data is generally unavailable for most songs; in this chapter, we enhance lyrical analysis by extensive emotion modeling for generating user recommendations without any user input.

2.2 Pre-processing for Word Representation Learning in Affective Tasks

In this section, we present an overview of related work on *pre-processing classification datasets* and *pre-processing word embeddings*, and how our work aims to bridge the gap between those efforts.

2.2.1 Pre-processing Classification Datasets

Pre-processing is a vital step in text mining and therefore, evaluation of pre-processing techniques has long been a part of many affective systems. Saif et al. [158] indicated that, despite its popular use in Twitter sentiment analysis, the use of a pre-compiled stoplist has a negative impact on the classification performance. Angiani et al. [11] analyzed various pre-processing methods such as stopwords removal, stemming, negation, emoticons, and so on, and found stemming to be most effective for the task of sentiment analysis. Similarly, Symeonidis et al. [170] found that lemmatization increases accuracy. Jianqiang and Xiaolin [85] observed that removing stopwords, numbers, and URLs can reduce noise but does not affect performance, whereas replacing negation and expanding acronyms can improve the classification accuracy.

Pre-processing techniques such as punctuation and negation Rose et al. [155] or pos-tagging and negation [160] make up a common component of many emotion classification models [95, 140]. One of the earliest works [39] preserved emotion words and negative verbs during stopwords removal, replaced punctuation with descriptive new words, replaced negative short forms with long forms, and concatenated negative words with emotion words to create new words (e.g., not happy $\rightarrow$ NOThappy). Although stemming may remove the emotional meaning from some words, it has been shown to improve classification accuracy [2, 39]. Negations have also been found beneficial,

whereas considering intensifiers and diminishers did not lead to any improvements [168].

Pecar et al. [141] also highlight the importance of pre-processing when using user-generated content, with emoticons processing being the most effective. Along the same lines, while Gratian and Haid [65] found pos-tags to be useful, Boiy et al. [20] ignored pos-tagging because of its effect of reducing the classification accuracy

The aforementioned works describe pre-processing techniques as applied directly to evaluation datasets in affective systems. In contrast, we examine the effectiveness of directly incorporating these known effective pre-processing techniques further "upstream" into the training corpus of word embeddings, which are widely used across a number of downstream tasks.

2.2.2 Pre-processing Word Embeddings

Through a series of extensive experiments, particularly those related to context window size and dimensionality, [105] indicate that seemingly minor variations can have a large impact on the success of word representation methods in similarity and analogy tasks, stressing the need for more analysis of often ignored pre-processing settings. Lison and Kutuzov [106] also present a systematic analysis of context windows based on a set of four hyperparameters, including window position and stopwords removal, where the right window was found to be better than left for English similarity task, and stopwords removal substantially benefited analogy task but not similarity.

A general space of hyperparameters and pre-processing factors such as context window size [77, 113], dimensionality [113], syntactic dependencies [104, 184] and their effect on NLP tasks including word similarity [77], tagging, parsing, relatedness, and entailment [73] and biomedical [37] has been studied extensively in the literature. The main conclusion of these studies, however, is that these factors are heavily task-specific. Therefore, in this work we explore pre-processing factors of generating word embeddings specifically tailored to affective tasks, which have received little attention.

A recent study investigated the role of tokenizing, lemmatizing, lowercasing and multiword grouping [26] as applied to sentiment analysis and found simple tokenization to be generally adequate. In the task of emotion classification, Mulik et al. [129] examined the role of four pre-processing techniques as applied to a vector space model based on tf-idf trained on a small corpus of tweets, and found stemming, lemmatization and emoji tagging to be the most effective factors.

Distinct from prior works, we examine a much larger suite of pre-processing factors grounded in insights derived from numerous affective systems, trained over two different corpora, using three different word embedding models. We evaluate the effect of the pre-processed word embeddings in three distinct affective tasks including sentiment analysis, emotion classification and sarcasm detection.

2.3 Affective and Contextual Embedding for Affect Detection

In this section, we present the related work on sarcasm detection including models that use *affective features*, *contextual information*, or a combination of both *affective features* and *contextual information*. We also explain how our work aims to bridge the gap among existing efforts.

2.3.1 Affective Features

Identifying sarcasm in text has evolved from simple lexical-based and syntactic pattern models [40, 63, 177] to complex models that consider refined linguistic features, such as positive predicates, interjections, gestural cues (emoticons, quotation marks, etc.) [29] or behaviour modeling [5, 151]. With the advent of deep learning, there has been a shift in how prediction models are designed and engineered. For example, Ghosh et al. [59] proposed a conditional LSTM network [79] for detecting sarcasm in twitter using sentence-level attention mechanisms on hashtags and [76] learned and utilized a knowledge-based model of affective features based on a wide range of lexical resources. These models mostly relied on manually engineered patterns and features. However, more recently, Zhang et al. [204] proposed multiple deep learning models, including a sentiment augmented/supervised with attention Bi-LSTM model and a sentiment transferred Bi-LSTM model to identify sarcasm in Twitter datasets.

The main drawback of these models is that they are based on self-contained content (e.g. Twitter hashtags #) and when such content is not available (which is the case for learning a more general model), the model fails to detect sarcasm. In addition, these models assume availability of the complete conversational context to detect sarcasm, which in most cases is not available; as a result they can not generalize properly. Distinct from this line of work, our proposed models minimize the need for manual feature engineering by utilizing an architecture with Bi-LSTM and an attention mechanism that can accurately learn the affective representation of an input, even by utilizing a single affective feature.

2.3.2 Contextual Features

To address issues of lack of generalization and manual feature engineering, general word representation learning models have been proposed, such as Word2vec [116] and GloVe [142]. These models learn embeddings of words that would locate them close to one another in the embedded space if they share common contexts in the corpus. However, this means that even words opposite in meaning to another (*antonyms*), such as *happy* and *sad* that often occur in similar contexts, would be embedded close to each other. As a result, the use of general models can be inadequate for affective tasks [10]. To this end, Zhang et al. [202] proposed a bi-directional gated recurrent neural network (GRNN) to capture syntactic and semantic information locally, and a pooling neural network to extract

contextual features automatically for sarcasm detection. While, Ilič et al. [84] proposed a deep learning model based on character-level word representations obtained from ELMo [143] using a learned representation that models features derived from morpho-syntactic cues to solve the issue of dissimilarity in context for sarcasm detection. In [189] a system based on a densely connected LSTM network with multi-task learning strategy using POS tag features was proposed. More recently, transformer-based models (i.e., using encoder and decoder methods), such as BERT [45], RoBERTa [107] and XLNet [198] have been proposed to advance the state of the art in many NLP tasks by combining word embeddings with context embedding by using attention mechanisms in a bidirectional manner. However, little work has considered using these models for sarcasm detection. A multi-modal sarcasm detection method including text, speech and video features was proposed, where pre-trained BERT-based model was used for representing the sentences [30]. Moreover, Potamias et al. [149] proposed RCNN-RoBERTa for sarcasm detection in social media by leveraging the pre-trained embeddings from RoBERTa combined with a recurrent convolutional neural network. Babanejad et al. [12] investigated the impact of data pre-processing on word representation learning for affective tasks.

The main drawback of using advanced pre-trained models is that they do not incorporate any affect-specific features during the training phase of the model – therefore the accuracy of the model on affective tasks can be lessened. Our proposed models incorporate affective features along with contextual information in one architecture for

sarcasm detection.

2.3.3 Affective-Contextual Features

More sophisticated approaches for sarcasm detection have tried to combine the best of the two worlds by incorporating affective features with general embedding models during training. For example, Felbo et al. [53] proposed DeepMoji by training a Bi-LSTM model with emojis to learn representations of emotional context. Through a series of extensive experiments, particularly those related to incorporating affective features with pre-trained embeddings for sarcasm detection, the authors demonstrated the need to consider affective features in word embedding models for sarcasm detection. Poria et al. [148] also developed pre-trained sentiment, emotion and personality models for identifying sarcastic text using Convolutional Neural Networks (CNN-SVM). More recently, [174] utilized a multi-dimension intra-attention mechanism to overcome limitations of sequential neural network models and capture words' incongruities for sarcasm detection. Agrawal and An [3] incorporated affective information into word representations by training a Bi-LSTM using corpora with weak affect labels and used such representations for sarcasm detection. Moreover, a model that uses the semantic, sentiment and punctuation based hand-crafted features for sarcasm detection was proposed using multi-head attention based Bidirectional Long-Short Term Memory (MHA-BiLSTM) with Glove pre-trained embeddings [99]. Finally, Hazarika et al. [74] proposed a ContextuAl SarCasm DEtector

(CASCADE), by adopting a hybrid approach of both content and context-driven modeling for sarcasm detection where they utilized a user's personality features and style of writing to detect sarcasm.

Our research is mostly related to this line of work. In particular, we advance the state of the art in sarcasm detection by combining recently proposed transformer-based language models, such as BERT and SBERT, with affective-specific features.

2.3.4 Task-specific Corpora for Sarcasm Detection

Another important observation for the problem of sarcasm detection is that using task-specific corpora can be beneficial [152]. Previous studies [86, 206] have employed datasets related to comedy (movie transcripts, novels, etc.) to improve on the sarcasm detection task. This is because the utterance of humor/emotion and sarcasm is more expressed in the comedy genre than in other genres. Comedy is a literary genre and a type of dramatic work that is often satirical in its tone. For instance, they used corpus of children's stories (e.g., Harry Potter Books), transcripts of a MTV show (e.g., Big Bang Theory) and transcripts of comedy TV series (e.g., Friends) to train models for emotion and sarcasm detection. In addition, when we are dealing with word embedding models such as BERT which is pre-trained by the universal language Wikipedia, leaving the usage of task-specific corpora challenges unresolved [194]. Deu et al., [49] proposed a model to solve this limitation by injecting the target domain data to BERT and encourage BERT to

be domain-aware for the task of sentiment analysis. In particular, the authors conducted post-training and adversarial training to improve the performance of Bert for sentiment classification. Following this intuition and to adhere to best practices, our proposed models leverage task-specific data of two sarcasm-rich corpora for training embeddings during training to improve the sarcasm detection accuracy.

3 Leveraging Emotion Features in Social Media Recommendations

3.1 Introduction

Recommender systems (RS) have widely and successfully been employed in domains as diverse as news and media, entertainment, e-commerce and financial services, to name a few. The main utility of such systems is their ability to suggest items to users that they might like or find useful. Traditionally, research on recommendation algorithms has focused on improving the accuracy of predictive models based on a combination of descriptive features of the items and users themselves (e.g., user behavior, interests and preferences) and the history of a user's interactions with the items through ratings, reviews, clicks and more [92, 136, 137]. However, little attention has been paid to the *emotional context* and its relation to recommendations.

While emotions can be manifested in various ways, we focus on emotions expressed in *textual information* that is associated with *items* or *users* in the system. For example,

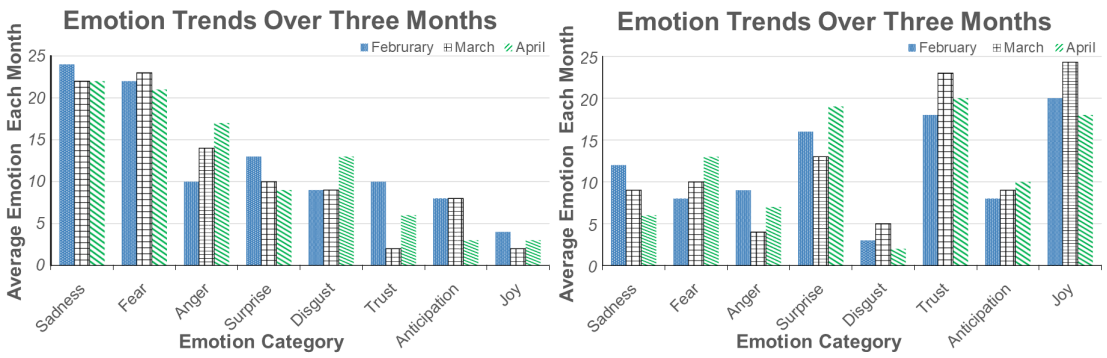

Figure 3.1: Illustrative example of emotions expressed in articles read by two different users, U_1 (left) and U_2 (right), over a three month period. Can we leverage the emotional context to improve recommendations?

the content of a news article, the content of an online review or the lyrics of a song are good examples of textual information *directly* associated with an item's emotional context. On the other hand, the *emotional profile* of a user can be determined through *explicit* or *implicit* feedback of users to items. Explicit feedback, such as providing ratings and/or submitting reviews to items, can represent an accurate reflection of a user's opinion about the item, but it is considered an intrusive process that disrupts the user-system interaction and negatively impacts user experience [135]. In addition, while it might be available for certain domains (e.g, product recommendations [34], movie recommendations [131], etc.), it is not easily obtainable in domains such as news or music, where users typically interact with items at a fast pace and are less inclined to provide feedback. In the absence, sparsity or high cost of acquisition of explicit feedback, incorporating implicit feedback, which is generally abundant and non-intrusive, might

be beneficial. Therefore, we focus on *indirectly* capturing the emotional context of users' activity by monitoring their interactions with items over time. For instance, one can monitor the emotions expressed in lyrics of songs users are listening to, or the tone of the stories in news article users are reading. Effectively, this information can be used to model a user's historical or temporal emotional profile.

To further motivate this, consider Figure 3.1 that illustrates the emotional profiles of two users, U_1 and U_2, based on eight basic emotions, expressed in articles read by them over a period of three months. One can notice that emotions of *sadness* and *fear* are mostly expressed in the articles read by U_1 while other emotions, such as *joy* are less expressed. In addition, one can observe trends such as the expression of *anger* increasing over time. On the other hand, for U_2, the emotions of *joy* and *trust* are mostly expressed and other emotions, such as *disgust* are less expressed. Moreover, emotions of *fear* and *anticipation* are increasingly expressed in the articles read by this user. Although, the emotional tone derived from news articles read by a user cannot justify the personality and state of mind of the user, it can be considered as the taste or preference of the user, where it shows the type of articles they are more interested in. Inspired by these observations, recent advancement in methods for emotion detection and the success of emotion-aware recommendation algorithms, the main motivation of our research is to investigate *whether*, *how* and *to what extent* emotion features can improve the accuracy of recommendations.

The Problem. More formally, the recommendation task can be described as follows. Let

a set of m users $\mathcal{U} = \{u_1, u_2, ..., u_m\}$ and a set of n items $\mathcal{I} = \{i_1, i_2, ..., i_n\}$. Let us also assume that each user u_i has already interacted with a set of items $\mathcal{I}_{u_i} \subseteq \mathcal{I}$ (e.g., consumed news articles). Then, the problem is to accurately predict the probability p_{u_a,i_j} with which a user $u_a \in \mathcal{U}$ will like item $i_j \in \mathcal{I} \setminus \mathcal{I}_{u_a}$. The task can also take the form of recommending a set $\mathcal{I}_k \subseteq \mathcal{I} \setminus \mathcal{I}_{u_a}$ of k items that the user will find most interesting (top-k recommendations). Depending on the target domain the semantics of the recommendation task can be instantiated differently. For example, in the news domain, the task is that of recommending an unread article. While, in music, it can be instantiated as the chance of re-listening to a song.

Challenges & Approach. In order to evaluate the importance of the emotional context to recommendations, we had to incorporate emotional features [4, 156, 185] to state-of-the-art recommendation algorithms and evaluate their accuracy performance. Figure 3.2, shows a schematic diagram of the emotion-aware recommendation algorithm process we designed, which consists of three main stages: i) *feature engineering*, ii) *model training*, and iii) *blending & ensemble learning*. Each of these components, define a number of challenges that need to be addressed. During *feature engineering*, we had to generate a number of features attributed to both users and items. Emphasis was given in capturing the most important non-emotional and emotional features for the prediction task. Once features are extracted, off-the-shelf *feature selection* methods are employed to select a subset of them that are more relevant for use in model construction. During *model training*,

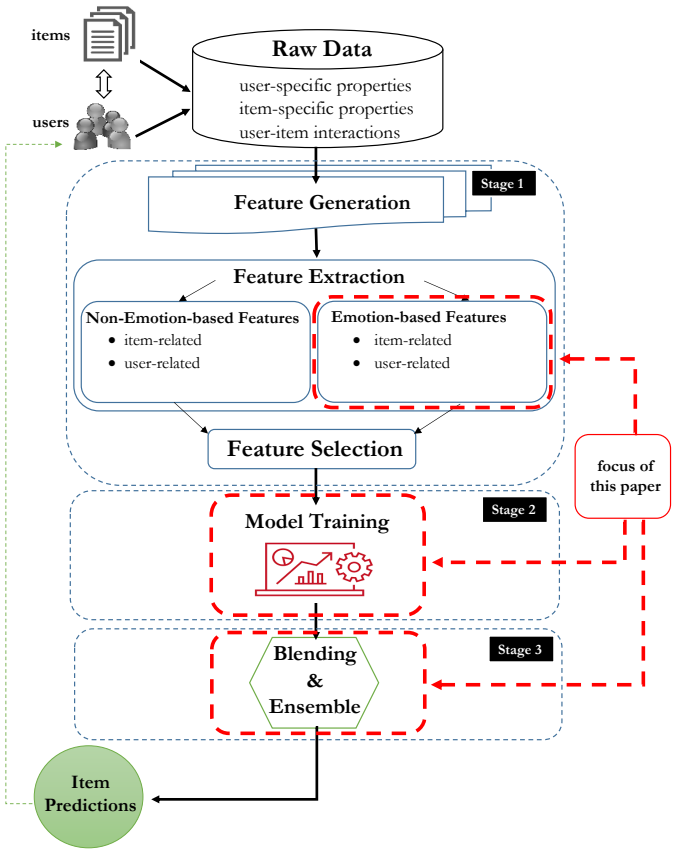

Figure 3.2: Overview of an emotion-aware recommendation system.

we experiment with a number of state-of-the-art models for generating recommendations. During *blending & ensemble* we combine alternative models to obtain better predictive models than any of the constituent models alone.

Contributions. The major contributions in this chapter are as follows:

- We systematically identify, extract and select the most relevant emotion-based features for use in recommendation models. These features are associated with both items and users in the domain. We supplement these features with typical

user and item features employed in recommendation tasks. Note that we had to limit the coverage to ones more suited to the targeted domains (news article, music). However, we are able to extract sufficient features to demonstrate the effect of emotions in recommendation accuracy.

- We devise a number of state-of-the-art models for generating recommendations that incorporate the additional emotion features. These models include variations of *gradient boosting decision trees*, *deep matrix factorization methods* and *deep neural network architectures*. In addition, we use ensembling methods to increase the predictive performance by blending or combining the predictions of multiple constituent models.

- We propose EMOREC, an emotion-aware recommendation model, which demonstrates the best accuracy performance in news recommendation task. EMOREC itself is an ensemble model.

- We conduct a thorough experimental evaluation on real datasets coming from diverse domains (news and music). Our results demonstrate that the emotion-aware recommendation models consistently outperform state-of-the-art non-emotion-based recommendation models. Our study provides evidence of the usefulness of the emotion features at large, as well as the feasibility of our approach on incorporating them to existing models to improve recommendations.

The remainder of this chapter is organized as follows. Section 3.2 describes the feature engineering process for selecting emotion and non-emotion features from item and users. In Section 3.3 we provide details of the proposed models for generating recommendations. Section 3.4 presents an experimental evaluation of the models and discusses the findings. Finally, Section 3.5 summarize this chapter.

3.2 Features for recommendation

This section describes the feature extraction procedure which is utilized in our proposed framework. The features are grouped into two main categories: (i) *emotion-based features* for items and users, and (ii) *non-emotion-based features* for items and users.

3.2.1 Emotion-based Features

The main objective of this chapter is to improve the performance of recommender system by leveraging the user/item emotion features.

Figure 3.3 shows an example of textual content of items (i.e., lyrics of a song, and news article) in music and news domains. As it can be observed, there are several words such as *laugh*, *love* and *gratifying*, expressing the emotion of *happiness*. Moreover, interjections such as *yay* and *oh* can be indicators of different emotions [61]. In this section, we describe how we extract such features to improve the recommendation system effectiveness. In order to maintain consistency, each news article is preprocessed by tokenizing into words,

excerpt of song lyrics

I need the emotion, I need to **laugh**
Sometimes I need to **cry**
I need the emotion. I need to **love** so hard
And **baby**, I need you
Oh, Sometimes I need a **good** old fashioned **fight**
It does not matter If I am **wrong** or **right**

excerpt of news article

"It was [**gratifying**] **yay**… Who wouldn't want it to keep going?" Walker told NBA.com. "**Oh**, I know teams will be gearing up on me and double-teaming me. But I just want to **win**, man. I want to get back to the playoffs any way possible. I don't care what I average the rest of the year."

Figure 3.3: Example emotions expressed in textual content

removing the stopwords and POS-tagging to extract nouns, verbs, adverbs and adjectives. In particular, we focus on two approaches for computing emotion features: sentiment analysis, which classifies text into *neutral, positive* and *negative* sentiments, and emotion analysis which categorizes text into emotions such as *happiness, sadness, anger, disgust, fear* and so on. Note that we extract emotion features for both users and items.

3.2.1.1 Item Emotion-based Features

Number of Emotion Words: This feature represents the number of words in an emotion lexicon (i.e., WordNet-Affect, see Table 1) that occur in the item (i.e., news article) more than once.

Ekman's Emotion Label: We count the number of emotion words occurring in the text document for each emotion type (Ekman's six emotion categories [50]) and then the text is assigned an emotion label with the highest number of emotion words appearing in the text. If more than one emotion category has the highest count, 0 is assigned to this

Resources	Size	Emotion Taxonomy
WordNet-Affect [167]	4787 words	Several
ISEAR [186]	7600 sentences	ISEAR
NRC [124]	14,182 words	Plutchik
SentiWordNet 3.0 [15]	11,000+ synset	Sentiments

Table 3.1: Emotion Resources

feature, leaving the next feature to indicate mixed emotions. A combination of different lexicons (WordNet-Affect and NRC, see Table 3.1) is used to find the emotion labels. We use multiple resources to have a bigger set of emotion words for each emotion.

Mixed Emotions: This feature indicates whether an item has more than one document-level emotion labels based on Ekman's emotion model (i.e., if two or more emotions have the highest score, this feature is valued at 1, otherwise 0). Since the initial annotation effort (previous feature) illustrated that in many cases, a sentence can exhibit more than one emotion, we have an additional category called mixed emotion to account for all such instances.

Sentiment Feature: The text is classified into three categories: *positive, negative* and *neutral*. We utilize the approach introduced in [133] and use SentiWordNet [15].

Interjections: This feature counts the number of interjections in a document. A short

sound, word or phrase spoken suddenly to express an emotion, e.g., oh, look out!, ah, are called interjections[1]. Our preliminary analysis found that interjections were common in quotes in news articles, which can be detected for potential emotions.

Capitalized Words: This feature counts the number of words in a document with all uppercase characters. People use capital words to express an emotion [177] and make it bold to the readers (e.g., *I said I am FINE*).

Punctuation: Two features are included to model the occurrence of question marks and exclamation marks repeated more than two times in a document. Using punctuation can clarify the emotional content of the texts that are sometimes easy to miss [177].

Grammatical Markers and Extended Words: This feature counts the number of times words with a character repeated more than two times (e.g., haaappy or oh yeah!!????) [22] as excessive use of letters in a word (e.g., repetition) is one way to emphasize feelings.

Plutchik Emotion Scores: First, we measure the semantic relatedness score between a word W_i in the text and an emotion category C_j in the NRC lexicon (see Table 1) as follows [2]:

$$PMI(W_i, C_j) = \sqrt[n]{\prod_{k=1}^{n} PMI(W_i, C_j^k)} \tag{3.1}$$

[1]List of interjections derived from: i) https://surveyanyplace.com/the-ultimate-interjection-list, ii) https://7esl.com/interjections-exclamations, and iii) https://www.thoughtco.com/interjections-in-english-1692798

where C_j^k ($k = 1 \ldots n$) is the k^{th} word of emotion category C_j. PMI is the Pointwise Mutual Information calculated as follows:

$$PMI(W_i, C_j^k) = \log \frac{P(W_i, C_j^k)}{P(W_i)P(C_j^k)} \tag{3.2}$$

where $P(W_i)$ and $P(C_j^k)$ are the probabilities that W_i and C_j^k occur in a text corpus, respectively, and $P(W_i, C_j^k)$ is the probability that W_i and C_j^k co-occur within a sliding window in the corpus. Finally, we calculate the average, maximum and minimum of score for all words in the text for each emotion category and consider each as an individual feature.

3.2.1.2 User Emotion-based Features

As we do not have access to users' explicit emotion towards items, we develop users' *implicit* emotional profile based on their historical interactions with items. By computing the emotion profile of the items with which a user is interacting, we derive the emotional taste of the user over that period of time over the set of items.

User Emotions Across Items: We determine the emotion score (i.e., Plutchik's emotion scores) for the last accessed item before subscription as well as for the last 20 items accessed by the user. Then, we pick the top 3 frequent emotions.

User Emotions Across Categories: We determine the emotion of categories of items (e.g., sports in news domain or pop in music) accessed by a user by counting the number

of items assigned to an emotion in a specific category, with the most frequent emotion considered as the emotion of the category. The feature is calculated for the whole history of the user.

3.2.2 Non-Emotion-based Features

Non-emotion-based features can also be classified into item-based and user-based features.

3.2.2.1 Item Non-Emotion-based Features

Item Topic: We extract topics in the article using Latent Dirichlet Allocation (LDA) [19]. In LDA, each topic is a distribution over words, and each document is a mixture of topics. The number of topics for the news articles are 112 , which were chosen empirically to minimize the perplexity score of the LDA result. Thus, the item topic is represented by a vector of length 112.

Topic Label: We use lda2vec [126] to generate and label the topics in an item (i.e., document), where each generated topic is labeled by one of its top k words which is most semantically similar to the other words in the top k word list. We then label the item (i.e., document) with the label of the most coherent topic among the top m topics of the document. The word vector of this label word is used as the value for this feature.

TF-IDF: This feature represents items as n-grams (unigram, bigram, trigram) with the TF-IDF weighting approach [110].

Coherence: We first calculate the cosine similarity scores between all pair of words in an item using word2vec pre-trained word vectors[2], and then record average of similarity scores, standard deviation of similarity scores, the lowest score that is higher than the standard deviation, and the highest score that is lower than the standard deviation as four features.

Potential to Trigger Subscription: This feature represents the total number of times the item was requested right before a paywall was presented to a user who subsequently made a subscription [42, 43]. In a subscription-based item delivery model a paywall is the page asking for subscription before allowing an unsubscribed user to continue accessing items.

3.2.2.2 User Non-emotion-based Features

Visit Count: We calculate the average number of items (articles) accessed by a user per visit. A visit is terminated if a user is inactive for more than 30 minutes.

User Spent Time: Two features are represented. One is the average time the user spent per item, and the other is the average time the user spent per visit.

User Interest in Subcategory: This feature represents the empirical probability of subcategory s given a user u and a category c denoted as $P(s|u,c)$. For example, $P(election|u, politics)$ can be determined by the total number of articles the user read on

[2]https://code.google.com/archive/p/word2vec/

Emotion Features	**Gain Score**
Plutchik emotion scores	3200.86
User emotions across items	1985.36
User emotions across categories	1850.33
Ekman's emotion label	1101.38
Punctuation	910.55
Grammatical markers and extended words	860.13
Interjections	773.12
Capitalized words	640.21
Mixed emotions	526.97
Sentiment features	360.68

Table 3.2: List of Emotion Feature Importance

election over the total number of articles that the user read on politics. In our experiments, the categories and subcategories were provided with the dataset and we consider only the top 50 most frequently visited subcategories for this feature.

User Latent Vector: We calculate the latent vector for each user based on matrix factorization introduced in [171]. This feature is chosen so that we can compare our method with the Deep Matrix Factorization model in [197], a state-of-the-art recommendation

Non-emotion Features	Gain Score
User latent vector	3640.87
Potential to trigger subscription	2974.46
User interest in subcategory	1530.28
Topic labeling	1421.19
User spent time	1110.57
Visit count	920.53
Item topic	867.12
Coherence	685.23
TF-IDF	410.29

Table 3.3: List of Non-emotion Feature Importance

method, which uses this feature as input for a deep neural network.

3.2.3 Feature Selection

One of the critical steps after feature extraction is to select important features for recommendation. Table 3.2 reports the most important emotion features and Table 3.3 non-emotion features according to gain importance score for the News data set. We evaluate feature importance by averaging over 10 training runs of a gradient boosting

machine learning model XGBoost [35] to reduce variance[3]. Also, the model is trained using early stopping with a validation set to prevent over-fitting to the training data. By using the zero importance function, we find features that have zero importance according to XGBoost.

3.3 Emotion-aware Recommendation Model (EmoRec)

In this section, we introduce a tailored structure of an Emotion-aware Recommender System Model (EMOREC) for personalized recommendation. Our final model is an ensemble model of three models leveraging both emotion/non-emotion-based features. The final model benefits from the strengths of the base models generalizing well over different application domains (i.e., news and music). We describe the structure of the proposed model and the training methods next.

3.3.1 Model Training

In this section we describe the details of each training models.

[3]Variance refers to the sensitivity of the learning algorithm to the specifics of the training data (e.g., the noise and specific observations).

3.3.1.1 Model 1 (Boost Model)

Gradient Boosting Decision Tree (GBDT) methods are among the most powerful machine learning approaches which have been effectively used in many domains [54] including recommendation [207]. The basic idea in GBDT approaches is to learn a set of *base/weak learners* (i.e., decision trees) sequentially by using different training splits. More precisely, at each step, we learn a new base model by fitting it to the error residuals (i.e., difference between the current model predictions and the actual target values) at that step. The new model outcome is the previous model outcome plus the (weighted) new base learner outcome. Eventually, the final model outcome is the weighted average of the outcomes of all base learners, where the weights are learned jointly with the base learners. We train two state-of-the-art GBDT models, namely, XGBoost [35] and Catboost [47], on our training datasets with the features selected in Section 3.2.3 as the input.

XGBoost uses pre-sorted/histogram-based algorithm to compute the best split while CatBoost uses *ordered boosting*, a permeation based algorithm, to learn the weak learners effectively. Moreover, XGBoost uses one-hot encoding before supplying categorical data, but CatBoost handles categorical features directly. We train both models individually (three base models for each). The final model output (i.e., probability that a user is interested in an item) is the combination of all base models outcomes:

$$\sum_{i}^{6} \alpha_i p_i \tag{3.3}$$

where p_i is the probability that the user is interested in the item according to base model i and α_i is the weight of base model i learned by XGboost/Catboost.

3.3.1.2 Model 2 (Deep Neural Network (DNN))

Figure 3.4 shows our proposed Deep Neural Network architecture for leveraging the emotion features (and other commonly available features) for the recommendation purpose. The input is divided into four groups[17]: i) user non-emotion based features, ii) item non-emotion based features, ii) user emotion-based features, and iv) item emotion-based features. For the categorical inputs, we utilize one-hot encoding (the second layer is look-up embeddings mapping each categorical feature to a fixed length embedding vector).

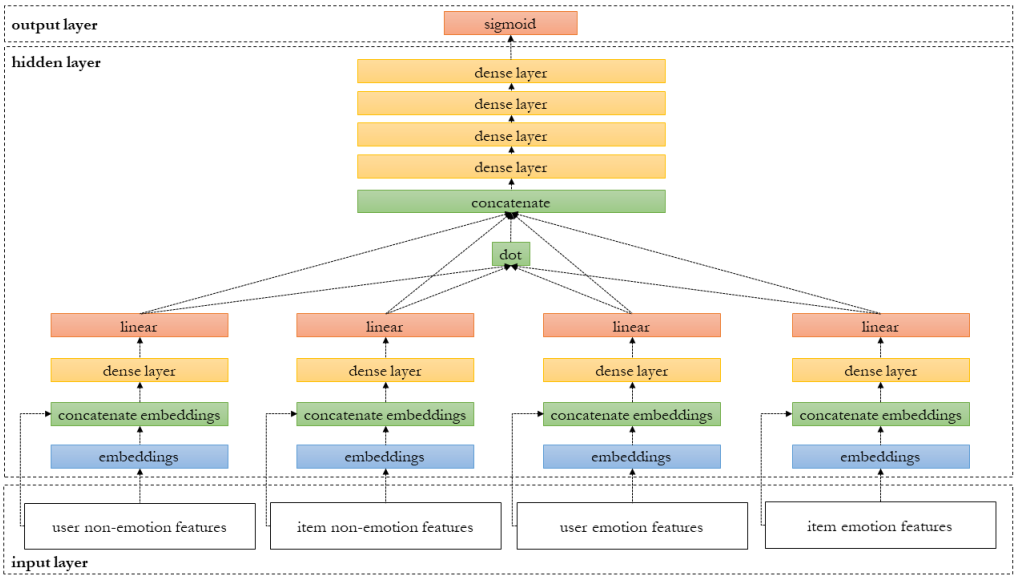

Figure 3.4: The Structure of Our DNN Model

In the architect "Dense Layer" can be formalized as:

$$Dense(x) = f(Wx + bias) \tag{3.4}$$

where W and $bias$ are parameters, x is the layer input and f is the activation function (for linear layer f is the identity function). We use L_2 regularization to prevent over-fitting in embedding layer and use back-propagation to learn the parameters.

3.3.1.3 Model 3 (Deep Matrix Factorization (Deep MF))

Inspired by the models proposed in [75, 197], we built our Deep MF (Figure 3.5) to leverage extra user/item features (i.e., emotion and non-emotion features) in the recommendation prediction task. In [197], they construct a user-item matrix with explicit ratings and implicit preference feedback, then with this matrix as the input, they present a deep neural architecture to learn a low dimensional space for the representation of both users and items. In [75], by replacing the inner product with a neural architecture, they learn an arbitrary function to capture the interactions between user and item latent vectors. Different from their work, we focused on modeling the user/item with rich extra features, such as non-emotion and emotion based features, as well as using embedding vectors learned in our DNN model. The input of our proposed model is the same as the DNN model where the categorical features are encoded using one hot vectors. The second

layer is the look-up embedding. In this layer, we have both MF embedding vectors, which we estimate through the learning process, and DNN embedding vectors, which are concatenation of embedding vectors (for each similar input group) learned from DNN model (they are fixed in this model). Generalized Matrix Factorization (GMF) layer combines two embeddings using dot product and applies some non-linearity. Similar to DNN model, the output of the model is the probability that a user is interested in an item.

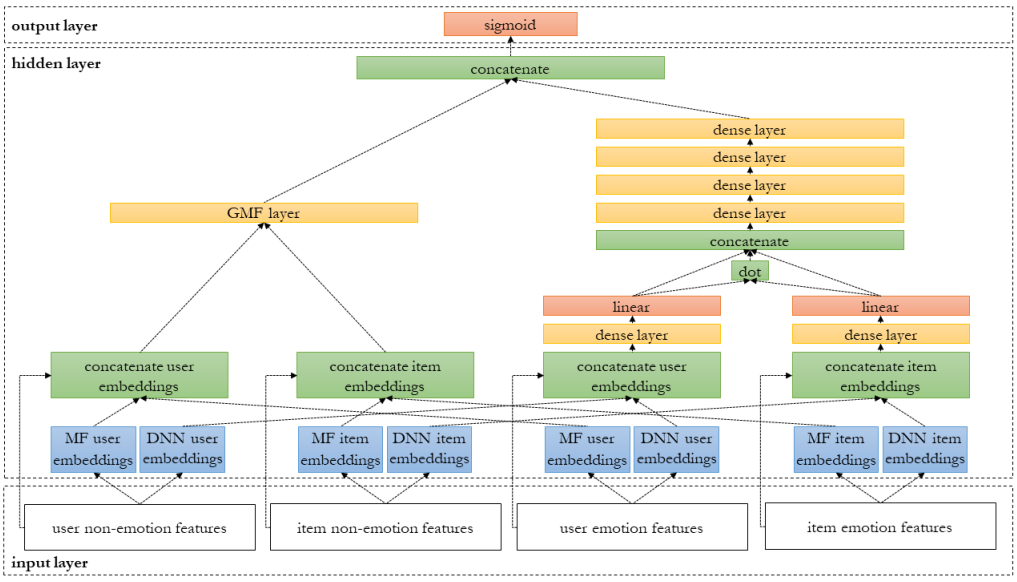

Figure 3.5: The Structure of Our Deep MF Model

Ensemble/Blending Model: The final model EMOREC was the weighted average of the three models' predictions. We use Nelder-Mead Method [134] to find the optimum weights of each models.

3.4 Experimental Evaluation

In this section, we introduce the data, evaluation protocols and the specific configurations used in our experiments.

3.4.1 Data

Our experiments are conducted on real-world datasets from two domains: news and music.

News: The Globe and Mail is one of the major newspapers[4] in Canada. We use the data spanning from January to July 2014 (a 6-month period) in our experiments where the data in the first 4 months were used for training, and the last 2 months for testing. The dataset contains information for 359,145 articles in total and 88,648 users in total, out of which 17,009 became subscribers during this period, and 71,639 were non-subscribers. Every time a user reads an article, watches a video or generally takes an action in the news portal, the interaction is recorded as a hit. Typically, a hit contains information like date, time, user id, visited article, special events of interest like subscription, sign in, and so on.

Music: This dataset originates from KKBOX[5] provided by Kaggle [6]. Our dataset comprises of 98,746 English songs and 34,403 users. Since the original dataset only

[4]https://www.theglobeandmail.com/

[5]https://www.kkbox.com

[6]https://www.kaggle.com/c/kkbox-music-recommendation-challenge

includes a song name, we supplement it by obtaining lyrics from musiXmatch[7] and MetroLyrics[8]. The dataset has 7,377,418 number of records from 2016-08 to 2017- 01 in the training set, and 2,556,790 records from 2017-01 to 2017-02 in the testing set which is time sensitive[9], we decided to use the last 20% of the training data for validation, and when we generated the predictions for the testing set.

3.4.2 Evaluation Metrics

We use F-score to measure the predictive performance of a recommender system. For each user in the test data set, we use the original set of read articles in the test period as the ground truth, denoted as T_g. Assuming the set of recommended news articles for the user is T_r, precision, recall, and F-measure are defined as follows:

$$Precision = \frac{|T_g \cap T_r|}{|T_r|}, \quad Recall = \frac{|T_g \cap T_r|}{|T_g|}$$

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The F-score on a test data set is the average over all the users in the test data set.

[7]https://labrosa.ee.columbia.edu/millionsong/musixmatch

[8]https://www.programmableweb.com/api/metrolyrics

[9]Note: Since the data in music dataset is ordered chronologically, we use the index as the timestamp to help the models for pattern evolution during the long period of time.

3.5 Discussion and Analysis

In this section, We analyze the impact of using various emotion features on the performance of recommendation models.

3.5.1 Comparing Recommendation Models with and without Emotion Features

Our main objective is to see whether the use of emotion features will boost the performance of recommendation models. For such a purpose we run the three state-of-the-art recommendation models described in the last section and some ensembles formed by these models with and without emotion features. The models used in our evaluation are as follows:

- *Single Boost Model*: We run XGBoost and Catboost separately to make predictions and collect the average of their F-scores.
- *Boost Blend*: This is the 6-model ensemble described in Model 1 in Section 3.3.1.
- *Deep MF*: This is the deep matrix factorization model described in Section 3.3.1.
- *Single DNN model*: We run the DNN model for 5 times with the same hyperparameters but different random seeds and collect the average result over 5 runs.
- *DNN Ensemble*: An ensemble of 5 DNN models with different hyperparameters (e.g., different learning rates, etc.) is run 5 times each with a different random seed.

The average result over the 5 runs is collected.

- *Boost Blend + Deep MF*: This is an ensemble consisting of Boost Blend and Deep MF.
- *Boost Blend + DNN Ensemble*: This an ensemble consisting of Boost Blend and DNN Ensemble.
- *Deep MF + DNN Ensemble*: This is an ensemble consisting of Deep MF and DNN Ensemble.
- *Boost Blend + Deep MF + DNN Ensemble*: an ensemble consisting of Boost Blend, Deep MF and DNN Ensemble.

We train each of the above models using the training data of our data set and use the trained model to make recommendations by predicting a user's interest in an item in the test data. Tables 3.4 and 3.5 show the results (in F-score) of using these recommendation methods with and without emotion features on the News and Music datasets accordingly, where the whole set of emotion features described in Section 3.2.3 is used in the results for "All", while none of the emotion features is used in the results for "Non-Emo". As can be seen, adding emotion features improves the predictive performance for all the recommendation methods. Among the single recommendation models (i.e., Single Boost Model, Deep MF and Single DNN Model), Deep MF performs the best. The results also show that ensemble methods perform better than single/component models. The best

Model	Non-Emo	All
Single Boost Model	70.19	70.86
Boost Blend	70.69	71.50
Deep MF	72.93	73.29
Single DNN Model	70.88	73.00
DNN Ensemble	73.62	74.30
Boost Blend + Deep MF	73.07	74.98
Boost Blend + DNN Ensemble	74.00	74.23
Deep MF + DNN Ensemble	74.61	75.10
EMOREC (Boost Blend + Deep MF + DNN Ensemble)	78.20	**80.30**

Table 3.4: Results of our Models on News Dataset (F-score)

performance is produced by the largest ensemble (i.e., *Boost Blend + Deep MF + DNN Ensemble*). We refer to this best-performing model as our EMOREC model. To further investigate whether there is a significant improvement of using "All" features over the use of no emotional features (“Non-Emo") on both news and music datasets, we conducted paired t-tests [101] on the results in Tables 3.4 and 3.5. The resulting p-value for the News dataset is 0.001874 and that for the Music dataset 0.002975, indicating a significant improvement on both datasets.

Model	Non-Emo	All
Single Boost Model	67.79	70.13
Boost Blend	71.08	70.61
Deep MF	70.00	71.00
Single DNN Model	71.30	72.29
DNN Ensemble	71.64	74.81
Boost Blend + Deep MF	70.00	70.03
Boost Blend + DNN Ensemble	72.01	74.87
Deep MF + DNN Ensemble	73.18	74.90
EMOREC (Boost Blend + Deep MF + DNN Ensemble)	73.68	**76.06**

Table 3.5: Results of our Models on Music Dataset (F-score)

3.5.2 Comparison with Other Baselines

We also compare our EMOREC model with the following three state-of-the-art recommendation methods with well-tuned parameters (that is, the parameters are optimally tuned to ensure the fair comparison). The objective is to investigate whether emotion features can smarten up these recommender systems. A brief description of these three models is as follows:

Basic MF: This is the simple matrix factorization model where used for discovering la-

tent features between two entities (i.e., user and articles)[171]. Both user preferences and item characteristics are mapped to latent factor vectors. Each element of the item-specific factor vector measures the extent to which the item possesses one feature. Accordingly,each element of the user-specific factor vector measures the extent of the user preferences in that feature.

FDEN and GBDT: an ensemble of different models, including Field-aware Deep Embedding Networks and Gradient Boosting Decision Trees [17]. The predictions of FDENs are from a bagging ensemble using the arithmetic mean of many networks, each of which has slight differences on hyper-parameters, including the forms of the activation.

Truncated SVD-based Feature Engineering: a gradient boosted decision trees model with truncated SVD-based embedding features [161]. To overcome the cold start problem, a truncated SVD-based embedding features were created using the embedding features with four different statistical based features (users, items, artists and time), the final model was the weighted average of the five models' predictions.

The results are illustrated in Table 3.6, which shows that emotion features can also improve the recommendation performance of these three state-of-the-art baselines. In addition, our EMOREC model performs significantly better than these three baselines in both cases of using emotion features and not using emotion features.

Model	Non-Emo	All
Basic MF	69.10	71.23
FDEN and GBDT	72.02	73.28
Truncated SVD-based Feature Engineering	73.12	74.01
EMOREC	78.20	**80.30**

Table 3.6: Comparison of EMOREC with State-of-the-art Baselines on News Dataset (F-score)

Model	Non-Emo	All
Basic MF	69.10	71.23
FDEN and GBDT	70.52	71.20
Truncated SVD-based Feature Engineering	71.98	72.54
EMOREC	73.68	**76.06**

Table 3.7: Comparison of EMOREC with State-of-the-art Baselines on Music Dataset (F-score)

3.5.3 Effect of Individual Emotion Features

Table 3.8 presents the results of a feature ablation study in order to further understand the effect of individual emotion features used in EMOREC.

In each run of this study, we keep all the features except one type of emotion features. The results indicate that removing Plutchik emotion scores (item feature), User emotions across categories and User emotions across items (user features) lead to considerable decline in the performance. It also shows that our model is able to capture useful implicit user emotion effectively.

To further validate the effectiveness of the top emotion features as learned from our experiments, we run a further experiment incorporating only the top three emotion features (i.e., Plutchik emotions, User emotions across categories, and User emotions across items) on six state-of-the-art recommendation models. As the results in Table 3.9 show, only using these three emotion features can also improve the recommender systems, with Basic MF showing the most gain.

Emotion Features	**News**	**Music**
ALL emotion features	80.30	77.03
- Sentiment features	78.15	76.66
- Mixed emotions	76.90	75.49
- Capitalized words	76.21	75.30
- Interjections	75.84	75.00
- Grammatical markers and extended words	75.23	74.94
- Ekman's emotion label	74.98	72.28
- Punctuation	75.17	73.10
- User emotions across categories	74.15	71.69
- User emotions across items	73.23	71.33
- Plutchik emotion scores	72.10	69.28

Table 3.8: Effect of Individual Emotion Features (F-score)

Model	No Emotion	Top Three Emotion
Basic MF	69.10	70.38
Boost Blend	70.69	71.00
FDEN and GBDT	72.02	72.77
Deep MF	72.93	73.01
Truncated SVD-based	73.12	73.60
DNN Ensemble	73.62	73.98

Table 3.9: Effect of Top Three Emotion Features (*Plutchik emotions, User emotions across categories, and User emotions across items*) on State-of-the-art Models

3.6 Summary

Motivated by the recent development in emotion detection methods (in textual information), we considered the problem of leveraging emotion features to improve recommendations. Towards that end, we derived a large number of emotion features that can be attributed to both items and users in news domain and can provide an emotional context. Then, we devised state-of-the-art *non-emotion* and *emotion-aware recommendation models* to investigate *whether*, *how* and *to what extent* emotion features can improve recommendations. To the best of our knowledge, this is the first attempt to systematically

and broadly evaluate the utility of a number of emotion features for the recommendation task. Our results indicate that emotion-aware recommendation models consistently outperform state-of-the-art non-emotion-based recommendation models. Furthermore, our study provided evidence of the usefulness of the emotion features at large, as well as the feasibility of our approach on incorporating them to existing models to improve recommendations.

As a more tangible outcome of the study, we proposed EMOREC, an emotion-aware recommendation model, which demonstrates the best predictive performance in news recommendation task. EMOREC itself is an ensemble model combining three models (*Boost Blend + Deep MF + DNN Ensemble*). It significantly outperforms other state-of-the-art recommendation methods evaluated in our experiments. We also evaluated the proposed emotion features individually. Among the emotion features examined, the Plutchik emotion scores of items (obtained by computing PMI scores between words) and user emotion profiles (based on the emotion scores of the items that the user accessed) are the most important.

4 A Comprehensive Analysis of Pre-processing for Word Representation Learning in Affective Tasks

4.1 Introduction

Affective tasks such as sentiment analysis, emotion classification and sarcasm detection have enjoyed great popularity in recent years. This success can be largely attributed to the fundamental and straightforward nature of the methods employed, the availability of vast amounts of user-generated natural language data, and the wide range of useful applications, spanning from hate speech detection to monitoring the sentiment of financial markets and news recommendation [13, 46]. Some recent models of affect analysis employed pre-trained word embeddings that have been obtained under the assumption of the distributional hypothesis [45, 115]. The distributional hypothesis suggests that two words occurring frequently in similar linguistic contexts tend to be more semantically similar, and therefore should be represented closer to one another in the embedding space. However, while such embeddings are useful for several natural language processing (NLP)

downstream tasks, they are known to be less suitable for affective tasks in particular [4, 172]. Although some authors claim that there is a need for post-processing word embeddings for affective tasks, others find that off-the-shelf vectors are very powerful for affective lexicon learning [106]. For example, `word2vec` [115] estimates the pair of words 'happy' and 'sad' to be more similar than the pair of words 'happy' and 'joy', which is counterintuitive, and might affect the accuracy performance of the models that depend on it.

To address the limitations of traditional word embeddings, several techniques have been proposed, including task-specific fine-tuning [45], retrofitting [52], representing emotion with vectors using a multi-task training framework [196] and generating affective word embeddings [53], to name a few. Other attempts to overcome the limitation of word vectors include optimization of hyperparameters [105], as well as fine-tuned pre-processing strategies tailored to different NLP tasks. While these strategies have demonstrated evidence of improving the accuracy performance in tasks such as word similarity, word analogy, and others [106], their effect in affective tasks has not received considerable attention and remains less explored.

Our work is motivated by the observation that pre-processing factors such as stemming, stop-words removal and many others make up an integral part of nearly every improved text classification model, and affective systems in particular [39, 140]. However, little work has been done towards understanding the role of pre-processing techniques applied

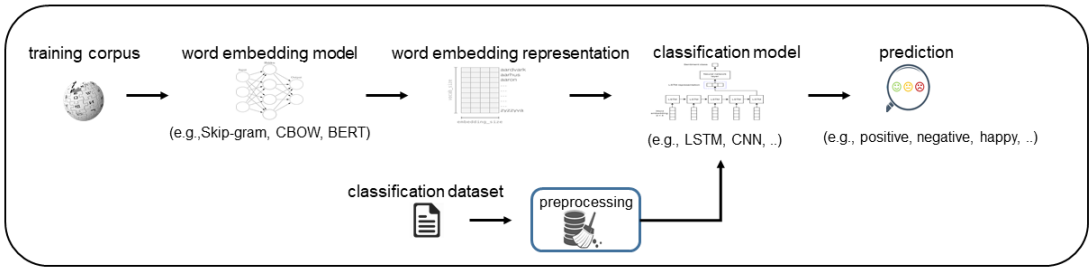

Figure 4.1: Framework of applying pre-processing in different stages in affective systems; (a) Pre, (b) Post.

to word embeddings in different stages of affective systems. To address this limitation, the overarching goal of this research, is to perform an extensive and systematic assessment of the effect of a range of linguistic pre-processing factors pertaining to three affective tasks, including *sentiment analysis*, *emotion classification* and *sarcasm detection*. Towards that end, we systematically analyze the effectiveness of applying pre-processing to large training corpora *before* learning word embeddings, an approach that has largely been overlooked by the community. We investigate the following research questions: *(i)* what is the effect of integrating pre-processing techniques earlier into word embedding models, instead of later on in a downstream classification models? *(ii)* which pre-processing techniques yield the most benefit in affective tasks? *(iii)* does pre-processing of word embeddings provide any improvement over state-of-the-art pre-trained word embeddings? and if yes, how much?

Figure 4.1 illustrates the difference between a) pre-processing word embeddings

pipeline (Pre) vs. b) pre-processing classification dataset pipeline (Post), where pre-processing techniques in (a) are applied to the training corpus of the model and in (b) only to the classification dataset. In brief, the main contributions of our work are as follows:

- We conduct a comprehensive analysis of the role of pre-processing techniques in affective tasks (including sentiment analysis, emotion classification and sarcasm detection), employing different models, over nine datasets;

- We perform a comparative analysis of the accuracy performance of word vector models when pre-processing is applied at the training phase (training data) and/or at the downstream task phase (classification dataset). Interestingly, we obtain best results when pre-processing is applied only to the training corpus or when it is applied to both the training corpus and the classification dataset of interest.

- We evaluate the performance of our best pre-processed word vector model against state-of-the-art pre-trained word embedding models;

- We make *source code* and *data* publicly available to encourage reproducibility of results[10]

The rest of this chapter is organized as follows: Section 4.2 elaborates on the pre-processing techniques employed in the evaluation of models. Section 4.3 describes the

[10]`https://github.com/NastaranBa/preprocessing-for-word-representation.`

experimental evaluation frame-work. In Section 4.4 a comprehensive analysis of the results is provided. Section 4.5 concludes the chapter with key insights of the research.

4.2 Pre-processing in Affective Systems

This section describes the pre-processing factors applied to the training corpus that is then used to generate word representations and the order of the pre-processing factors which we need to follow when applying on the corpus.

4.2.1 Pre-processing Factors

Basic: A group of common text pre-processing applied at the very beginning, such as removing html tags, removing numbers, and lower-casing. This step removes all common punctuation from text, such as "@%*=()/ +" using the NLTK regexptokenizer[11].

Spellcheck (`spell`): A case can be made for either correcting misspellings and typos or leaving them as is assuming they represent natural language text and its associated complexities. In this step, we identify words that may have been misspelled and correct them[12]. As unambiguous spell corrections are not very common and in most cases we have multiple options for correction, we built our own custom dictionary to suggest

[11] https://www.nltk.org/_modules/nltk/tokenize/regexp.html

[12] https://pypi.org/project/pyspellchecker/

a replacement by parsing the ukWac corpora[13] to retrieve a word-frequency list. A misspelled word that has multiple replacements is replaced with the suggested word that has the maximum frequency in the corpora.

Negation (`neg`): Negation is a mechanism that transforms a positive argument into its inverse rejection [18]. Specifically in the task of affective analysis, negation plays a critical role as negation words can affect the word or sentence polarity causing the polarity to invert in many cases. Our negation procedure is as follows:

(i) Compilation of an antonym dictionary: The first stage involves compiling an antonym dictionary using the WordNet corpus [117]. For every synset, there are three possibilities: finding no antonym, one antonym or multiple antonyms. The first two cases are trivial (unambiguous replacements). In the case of the third option (ambiguous replacement), which represents the most common case, amongst the many choices, we consider the antonym with the maximum frequency in the ukWac corpus, as described in the previous section and finally the antonym of a word is picked at random from one of its senses in our antonym dictionary.

(ii) Negation handler: Next, we identify the negation words in tokenized text[14]. If a negation word is found, the token following it (i.e., negated word) is extracted and its antonym looked up in the antonym dictionary. If an antonym is found, the negation word

[13]https://www.sketchengine.eu/ukwac-british-english-corpus/

[14]https://pypi.org/project/negspacy/

and the negated word are replaced with it.

For example, let the sentence "I am not happy today" in its tokenized form ['I', 'am', 'not', 'happy', 'today']. First, we identify any negation words (i.e., 'not') and their corresponding negated words (i.e., 'happy'). Then, we look up the antonym of 'happy' in the antonym dictionary (i.e., 'sad') and replace the phrase 'not happy' with the word 'sad', resulting in a new sentence "I am sad today".

Parts-of-Speech (`pos`): Four parts-of-speech classes, namely nouns, verbs, adjectives and adverbs have been shown to be more informative with regards to affect than the other classes. Thus, using the NLTK pos-tagger, for each sentence in the corpus we retain only the words belonging to one of these four classes, i.e., NN*, JJ*, VB*, and RB*.

Stop-words (`stop`): Stop-words are generally the most common words in a language typically filtered out before classification tasks. Therefore, we remove all the stop-words using the NLTK library.

Stemming (`stem`): Stemming, which reduces a word to its root form, is an essential pre-processing technique in NLP tasks. We use NLTK Snowball stemmer for stemming our training corpus.

4.2.2 Order of Pre-processing Factors

While some pre-processing techniques can be applied independently of each other (e.g., removing stop-words and removing punctuation), others need a more careful consideration

of the sequence in which they are applied in order to obtain a more stable result. For instance, pos-tagging should be applied before stemming in order for the tagger to work well, or negation should be performed prior to removing stop-words. To this end, we consider the following ordering when combining all the aforementioned pre-processing factors: spellchecking, negation handling, pos classes, removing stop-words, and stemming.

4.3 Experimental Evaluation

4.3.1 Training Corpora

Table 4.1 and §4.2 summarize the details of our two training corpora with regards to their vocabulary and corpus sizes after applying various pre-processing settings. For some pre-processing such as POS (`pos`) and stop-words removal (`stop`), without any significant loss in vocabulary as indicated by the % ratio of pre-processed to basic, the corpus size reduces dramatically, in some cases more than 50%, a non-trivial implication with regards to training time.

Processing	Vocab		Corpus	
	size	%	size	%
Basic	155K	100	123.2M	100
`spell`	149K	96	123.2M	100
`stem`	137K	88	123.2M	100
`punc`	147K	95	111.0M	90
`neg`	152K	98	90.7M	73
`stop`	150K	97	75.6M	61
`pos`	154K	99	70.7M	57

Processing	Vocab		Corpus	
	size	%	size	%
All - `punc`	151K	97	93.7M	76
All - `pos`	140K	90	90.5M	73
All - `stop`	150K	97	75.3M	61
All	110K	71	55.2M	49
All - `stem`	110K	71	58.1M	47
All - `spell`	110K	71	56.4M	46
All - `neg`	110K	71	54.3M	44

Table 4.1: Details of News training corpora

News: This corpus consists of 142,546 articles from 15 American publications, spanning from 2013 to early 2018[15].

Wikipedia: Comparatively a much larger corpus than the News, this corpus consists of 23,046,187 articles from Wikipedia [16].

[15]https://www.kaggle.com/ snapcrack/all-the-news

[16]https://www.kaggle.com/jkkphys/english-wikipedia-articles-20170820-sqlite

Processing	Vocab		Corpus	
	size	%	size	%
Basic	5.1M	100	8.1B	100
All - `punc`	4.9M	96	7.2B	89
All - `pos`	4.8M	94	7.0B	86
All - `stop`	4.9M	96	6.8B	84
All - `stem`	4.3M	84	6.4B	79
All - `spell`	4.6M	90	6.1B	75
All	4.6M	90	5.6B	69
All - `neg`	4.6M	90	5.0B	62

Table 4.2: Details of Wikipedia training corpora

4.3.2 Word Embedding Models

We obtain our pre-processed word representations through three models: (i) **CBOW (Continuous Bag-of-Words)**, (ii) **Skip-gram**: While CBOW takes the context of each word as the input and tries to predict the word corresponding to the context, skip-gram reverses the use of target and context words, where the target word is fed at the input and the output layer of the neural network is replicated multiple times to accommodate the chosen number of context words [115]. We train both the models on both the training corpora using min count of 5 for News and 100 for Wikipedia with window sizes of 5

Dataset	Genre	Task	Total
IMDB	reviews	sentiment	50,000
SemEval	tweets	sentiment	14,157
Airline	tweets	sentiment	11,541
ISEAR	narratives	emotions	5,477
Alm	fairy tales	emotions	1,206
SSEC	tweets	emotions	1,017
Onion	headlines	sarcasm	28,619
IAC	response	sarcasm	3,260
Reddit	comments	sarcasm	1,010,826

Table 4.3: Details of evaluation datasets

and 10, respectively, setting dimensionality to 300. (iii) **BERT (Bidirectional Encoder Representations from Transformers)**: BERT is an unsupervised method of pre-training contextualized language representations [45]. We train the model using BERT large uncased architecture (24-layer, 1024-hidden, 16-heads, 340M parameters) with same setting for parameters as the original paper.

We train each of the three models (CBOW, Skip-gram and BERT) 8 times using 16 TPUs (64 TPU chips), Tensorflow 1.15 with 1TB memory on Google Cloud and two 32-GPU clusters of V100/RTX 2080 Ti with 1TB memory using Microsoft CNTK

parallelization algorithm[17] on Amazon server. For a large model such as BERT, it takes up to 4-5 days for each run of the training.

4.3.3 Evaluation Datasets

We conduct our evaluation on three tasks, namely sentiment analysis, emotion classification and sarcasm detection. Table 4.3 presents the details of our evaluation datasets, and some illustrative examples of text are shown in Table 4.4.

Sentiment Analysis: This popular task involves classifying text as positive or negative, and we use the following three datasets for evaluation:

(i) **IMDB**: This dataset[18] includes 50,000 movie reviews for sentiment analysis, consisting of 25,000 negative and 25,000 positive reviews [109].

(ii) **Semeval 2016**: This sentiment analysis in Twitter dataset[19] consists of 14,157 tweets where 10,076 of them are positive and 4,081 negative [130].

(iii) **Airlines**: This sentiment analysis dataset[20] consists of 11,541 tweets about six U.S. airlines from February 2015, with 9,178 tweets labeled as positive and 2,363 negative.

[17]https://docs.microsoft.com/en-us/cognitive-toolkit/multiple-gpus-and-machines

[18]http://ai.stanford.edu/ amaas/data/sentiment/

[19]http://alt.qcri.org/semeval2016/task4/index.php

[20]https://www.kaggle.com/crowdflower/twitter-airline-sentiment

Dataset	Text	Label
IMDB	*I must admit that this is one of the worst movies I've ever seen. I thought Dennis Hopper had a little more taste than to appear in this kind of yeeeecchh... [truncated]*	negative
Airline	· *everything was fine until you lost my bag.*	negative
ISEAR	· *At work, when an elderly man complained unjustifiably about me and distrusted me.*	anger
Alm	· *The ladies danced and clapped their hands for joy.*	happy
SSEC	· *if this heat is killing me i don't wanna know what the poor polar bears are going through*	sadness
Onion	· *ford develops new suv that runs purely on gasoline*	sarcastic
IAC	· *Been saying that ever since the first time I heard about creationsism*	not-sarcastic
Reddit	· *Remember, it's never a girl's fault, it's always the man's fault.*	sarcastic

Table 4.4: Examples of text instances in the evaluation datasets

Emotion Classification: A multiclass classification task, which involves classifying text into a number of emotion categories such as happy, sad, and so on. The following datasets are used in our evaluation:

(i) **SSEC**: The Stance Sentiment Emotion Corpus [159] is the re-annotation of the SemEval 2016 Twitter stance and sentiment corpus [123] with emotion labels including

anger, joy, sadness, fear, surprise. [21].

(ii) **ISEAR**: This dataset contains narratives of personal experiences evoking emotions [186]. We use a subset of the data consisting of five categories: sadness, anger, disgust, fear, joy.

(iii) **Alm**: This dataset contains sentences from fairy tales marked with one of five emotion categories: angry-disgusted, fearful, happy, sad and surprised [31].

Sarcasm Detection: Detecting sarcasm from text, a challenging task due to the sophisticated nature of sarcasm, involves labeling text as sarcastic or not. We use the following three datasets:

(i) **Onion**: This news headlines dataset [22] collected sarcastic versions of current events from The Onion and non-sarcastic news headlines from HuffPost [119], resulting in a total 28,619 records.

(ii) **IAC**: A subset of the Internet Argument Corpus [132], this dataset contains response utterances annotated for sarcasm. We extract 3260 instances from the general sarcasm type.[23].

(iii) **Reddit**: Self-Annotated Reddit Corpus (SARC)[24] is a collection of Reddit posts where sarcasm is labeled by the author in contrast to other datasets where the data is

[21]SSEC: http://www.romanklinger.de/ssec/

[22]https://github.com/rishabhmisra/News-Headlines-Dataset-For-Sarcasm-Detection

[23]https://nlds.soe.ucsc.edu/sarcasm2

[24]SARC v0.0: https://nlp.cs.princeton.edu/SARC/0.0/

typically labeled by independent annotators [93].

4.3.4 Classification Setup

For classification, we employ the LSTM model as it works well with sequential data such as text. For binary classification, such as sentiment analysis and sarcasm detection, the loss function used is the binary cross-entropy along with sigmoid activation:

$$\xi = -\frac{1}{N}\sum_{i=1}^{N} y_i log(p(y_i)) + (1 - y_i) log(1 - p(y_i)) \tag{4.1}$$

where y is the binary representation of true label, $p(y)$ is the predicted probability, and i denotes the i^{th} training sample.

For multiclass emotion classification, the loss function used is categorical cross-entropy loss over a batch of N instances and k classes, along with softmax activation:

$$\xi = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{k} y_{ij} log\left(p(y_{ij})\right) \tag{4.2}$$

where $p(y)$ is the predicted probability distribution, $p(y_{ij}) \in [0, 1]$.

The optimizer is Adam [96], all loss functions are sample-wise, and we take the mean of all samples (epoch = 5, 10, batch size = 64, 128). All sentiment and sarcasm datasets are split into training/testing using 80%/20%, with 10% validation from training. For the smaller and imbalanced emotion datasets, we use stratified 5-fold cross-validation.

We use a dropout layer to prevent overfitting by ignoring randomly selected neurons during training. We use early stopping when validation loss stops improving with patience = 3, min-delta = 0.0001. The results are reported in terms of weighted F-score (as some emotion datasets are highly imbalanced), where F-score = $2\frac{p.r}{p+r}$, with p denoting precision, and r is recall.

4.4 Discussion and Analysis

We analyze the impact of pre-processing techniques in word representation learning on affect analysis.

4.4.1 Effect of Pre-processing Factors

A primary goal of this work is to identify the most effective pre-processing factors for training word embeddings for affective tasks. While, Table 4.5 and §4.6 details the results of our experiments comparing the performance of individual pre-processing factors as well as those of ablation studies (i.e., including all the factors but one) on News corpus, Table 4.7, §4.8 and §4.9 demonstrates the results of our experiments on Wikipedia corpus.

Observing the performance of the individual factors on the News corpus, we note that even a single simple pre-processing technique can bring improvements, thereby validating our intuition of incorporating pre-processing into training corpora of word representations. Second, negation (`neg`) processing appears to be consistently the most effective factor

Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
Basic	83.99	55.69	60.73	65.74	68.23	59.42	36.81	55.43	51.76
`stop`	84.43	55.72	61.37	66.03	68.17	59.27	36.81	56.01	52.33
`spell`	86.20	55.93	61.96	66.00	69.57	60.00	36.88	56.41	52.14
`stem`	86.92	55.72	61.86	65.89	68.49	59.72	36.94	55.84	51.89
`punc`	86.99	56.41	62.08	65.93	69.85	60.28	36.94	56.89	52.03
`pos`	85.66	56.83	62.75	66.32	70.25	60.63	37.02	57.04	53.19
`neg`	88.98	57.29	63.81	66.87	71.12	60.91	37.22	57.39	54.15
All	89.96	57.82	64.58	67.23	70.90	60.84	37.43	57.72	53.71
All - `neg`	84.67	55.00	61.58	66.02	69.73	59.94	36.91	55.89	51.94
All - `pos`	85.69	56.31	64.29	66.97	70.48	60.15	37.19	56.27	52.16
All - `punc`	86.41	56.88	63.01	66.75	70.01	60.00	37.01	57.19	52.43
All - `spell`	88.23	56.41	63.87	67.23	70.83	60.27	37.22	57.41	53.41
All - `stop`	**90.01**	**60.82**	66.84	67.20	**72.49**	**62.11**	**38.96**	**59.28**	55.00
All - `stem`	88.12	**60.82**	**67.12**	**69.25**	72.13	61.73	38.00	59.00	**55.42**

Table 4.5: F-score results of evaluating the effect of pre-processing factors using CBOW on News corpus. The overall best results are in **bold**. The best result using only any one pre-processing setting is underlined.

Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
Basic	83.07	54.23	61.47	65.51	68.01	59.75	35.87	55.64	51.49
`stop`	83.23	55.47	62.00	65.62	68.00	59.84	35.94	55.76	51.62
`spell`	85.90	55.48	62.00	65.61	69.76	60.28	36.10	55.93	52.30
`stem`	86.00	55.33	61.89	65.60	68.72	59.50	36.00	55.69	51.40
`punc`	86.68	55.79	62.38	65.89	70.00	60.44	36.41	56.81	52.71
`pos`	85.91	56.28	63.25	66.24	69.81	60.85	36.44	56.23	52.94
`neg`	87.28	56.89	63.72	66.87	70.59	61.27	36.87	57.34	53.10
All	88.36	57.04	64.91	66.94	70.73	61.12	37.10	57.92	53.58
All - `neg`	83.26	54.00	61.95	66.00	69.88	60.00	36.94	55.97	51.89
All - `pos`	86.21	55.22	65.12	66.06	69.88	61.00	37.00	56.42	52.10
All - `punc`	85.57	55.99	64.29	66.29	70.00	60.98	37.01	57.02	52.53
All - `spell`	86.00	56.98	65.00	66.25	70.25	0.61	37.04	57.69	52.86
All - `stop`	**88.74**	**60.93**	67.00	68.57	**72.20**	62.02	**38.92**	59.18	55.18
All - `stem`	88.42	60.67	**67.39**	**69.08**	72.00	**62.36**	37.44	**59.48**	**55.23**

Table 4.6: F-score results of evaluating the effect of pre-processing factors using Skip-gram on News corpus. The overall best results are in **bold**. The best result using only any one pre-processing setting is underlined.

Models	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
CBOW	Basic	84.91	56.89	68.11	69.15	71.02	63.58	45.22	59.73	55.84
	All	88.41	60.25	71.39	71.57	73.61	65.27	48.81	62.48	57.42
	All - `neg`	83.02	56.03	69.28	69.55	70.25	64.18	46.00	60.42	55.93
	All - `pos`	85.69	57.21	71.00	70.08	72.29	64.82	47.53	62.28	56.25
	All - `punc`	84.00	57.36	70.46	70.01	72.02	65.00	47.68	61.84	56.64
	All - `spell`	86.19	58.26	70.98	70.59	72.85	65.00	47.29	61.63	57.00
	All - `stop`	**91.10**	61.00	73.00	72.31	74.50	68.20	**52.39**	64.29	58.46
	All - `stem`	88.76	**62.19**	**73.25**	**72.36**	**75.69**	**68.53**	50.28	**65.33**	**59.28**

Table 4.7: F-score results of evaluating the effect of pre-processing factors using CBOW on Wikipedia corpus. The overall best results are shown in **bold**.

across all the 9 datasets, indicating its importance in affective classification, followed by parts-of-speech (`pos`) processing where we retained words belonging only to one of four classes. On the other hand, removing stop-words (`stop`), spellchecking (`spell`) and stemming (`stem`) yield little improvement and mixed results. Interestingly, applying all the pre-processing factors is barely better or in some cases even worse (Onion, Reddit and SSEC) than applying just negation. Finally, the best performance comes from combining all the pre-processing factors except stemming (All-`stem`). Moreover, the performance of ablation studies on Wikipedia corpus for all three models where we note that the best performance for the CBOW model comes from combining all the pre-processing factors

Models	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
Skip-gram	Basic	84.00	55.94	68.36	69.20	71.68	63.74	45.01	59.45	55.62
	All	87.00	59.99	71.29	71.25	73.82	65.67	48.51	**65.02**	57.13
	All - `neg`	84.97	56.11	69.00	70.17	70.04	64.55	46.28	60.54	55.86
	All - `pos`	86.21	57.62	70.25	70.85	73.22	65.47	47.49	63.44	56.00
	All - `punc`	85.00	57.20	70.00	70.77	72.00	65.00	47.10	61.72	56.49
	All - `spell`	85.75	58.49	70.26	70.89	72.63	65.18	47.14	61.25	56.84
	All - `stop`	**89.76**	**61.74**	72.19	**72.00**	**75.69**	68.29	**52.01**	64.00	58.14
	All - `stem`	89.66	60.28	**73.66**	71.98	75.24	**68.72**	51.39	63.44	**59.01**

Table 4.8: F-score results of evaluating the effect of pre-processing factors using Skip-gram on Wikipedia corpus. The overall best results are shown in **bold**.

except stemming (All-`stem`), whereas for the Skip-gram and BERT models, the best results are obtained by applying all the pre-processing factors except stop-words removal (All-`stop`). Considering that the Wikipedia corpus is almost 160 times bigger than the News corpus, it is unsurprising that the word embeddings obtained from the former yield considerably better results, consistent across all nine datasets.

4.4.2 Evaluating Pre-processing Training Corpora vs. Pre-processing Classification Dataset

We investigate the difference between applying pre-processing to the training corpora for generating word embeddings (Pre) and applying pre-processing to the classification

Models	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
BERT	Basic	90.11	70.82	90.23	71.19	76.30	59.74	57.81	65.70	65.39
	All	91.86	71.76	91.73	73.66	78.72	62.60	59.74	67.80	67.49
	All - `neg`	90.33	70.52	91.04	72.00	77.07	61.44	58.14	66.59	66.10
	All - `pos`	91.01	71.20	91.66	73.31	78.45	62.04	59.01	66.25	68.13
	All - `punc`	91.59	71.50	91.60	73.18	78.54	62.27	59.60	67.25	67.27
	All - `spell`	91.78	71.13	91.34	73.02	78.40	62.00	59.44	67.21	67.30
	All - `stop`	**94.18**	**73.81**	**94.85**	**75.80**	**79.10**	**65.39**	**60.73**	**69.33**	**69.81**
	All - `stem`	92.19	71.94	92.03	74.49	77.93	63.74	60.16	68.00	67.05

Table 4.9: F-score results of evaluating the effect of pre-processing factors using BERT on Wikipedia corpus. The overall best results are shown in **bold**.

datasets (Post). As an example, during *Pre*, we first apply the pre-processing techniques (e.g., all but stemming) to the training corpus (e.g., Wikipedia), then generate word embeddings, then convert a classification dataset (e.g., IMDB) into word embedding representation, and finally classify using LSTM. Conversely, for *Post*, we first generate word embeddings from a training corpus (e.g., Wikipedia), then apply the pre-processing techniques (e.g., all but stemming) to the classification dataset (e.g., IMDB), which is then converted to word vector representation, and finally classified using LSTM [25].

[25]Note: For settings including stemming, the classification data is also stemmed in order to obtain a compatible vocabulary.

Models	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
	Post	87.49	59.33	71.28	69.87	74.20	67.13	47.19	62.00	56.27
CBOW	Pre	**88.76**	62.19	**73.25**	**72.36**	**75.69**	68.53	50.28	**65.33**	*59.28*
	Both	88.10	**62.41**	73.00	71.86	75.00	**70.10**	**50.39**	64.52	58.20
	Post	88.14	60.41	71.85	70.22	75.07	67.00	50.44	62.08	56.00
Skip-gram	Pre	**89.76**	**61.74**	72.19	**72.00**	**75.69**	68.29	**52.01**	64.00	**58.14**
	Both	89.33	61.25	**73.58**	71.62	75.48	**68.74**	51.68	**65.29**	58.03
	Post	94.58	70.25	92.35	74.69	77.10	63.38	58.40	68.20	67.17
BERT	Pre	94.18	**73.81**	**94.85**	**75.80**	**79.10**	**65.39**	**60.73**	**69.33**	**69.81**
	Both	**94.63**	72.41	93.00	75.19	78.69	65.17	60.33	69.06	68.43

Table 4.10: F-score results of evaluating the effect of pre-processing word embeddings training corpus vs. pre-processing evaluation datasets

The results of this experiment are presented in Table 4.10, where we observe that incorporating pre-processing into the training corpora before generating word vectors (Pre) outperforms pre-processing classification datasets (Post) across all nine datasets of the three affective tasks. Interestingly though, pre-processing both the bodies of text (Both) appears to be of little benefit, suggesting the importance of pre-processing training corpora used for obtaining word embeddings.

Models	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
GloVe	85.64	70.29	70.21	70.19	71.39	63.57	56.21	65.30	58.40
SSWE	80.45	69.27	78.29	64.85	52.74	50.73	51.00	54.71	52.18
FastText	75.26	68.55	70.69	55.74	58.29	59.37	52.28	25.40	53.20
DeepMoji	69.79	62.10	71.03	65.67	70.90	53.08	46.33	58.20	58.90
EWE	71.28	60.27	67.81	67.43	70.06	55.02	58.33	66.09	58.94
Our best results:									
CBOW	91.10	62.19	73.25	72.36	75.69	68.53	52.39	65.33	59.28
Skip-gram	89.76	61.74	73.66	72.00	75.69	**68.72**	52.01	65.02	59.01
BERT	**94.18**	**73.81**	**94.85**	**75.80**	**79.10**	65.39	**60.73**	**69.33**	**69.81**

Table 4.11: F-score results of comparing against state-of-the-art word embeddings. The best score is highlighted in **bold**, and the second best result is underlined.

4.4.3 Evaluating Proposed Model against State-of-the-art Baselines

While not a primary focus of this work, in this final experiment we compare the performance of our pre-processed word embeddings against those of six state-of-the-art pre-trained word embeddings[26].

(i) **GloVe**: Global vectors for word representations [142] were trained on aggregated global word co-occurrences. We use the vectors trained on GloVe6B 6 billion words[27],

[26]These vectors obtained from their original repositories have been used without any modifications.

[27]https://nlp.stanford.edu/projects/glove/

uncased, from Wikipedia and Gigaword.

(ii) **SSWE**: Sentiment Specific Word Embeddings (unified model)[28] were trained using a corpus of 10 million tweets to encode sentiment information into the continuous representation of words [172].

(iii) **FastText**: These pre-trained word vectors[29], based on sub-word character n-grams were trained on Wikipedia using fastText [21], an extension of the word2vec model.

(iv) **DeepMoji**: These word embeddings[30] were trained using BiLSTM on 1.2 billion tweets with emojis [53].

(v) **EWE**: Emotion-enriched Word Embeddings[31] were learned on 200,000 Amazon product reviews corpus using an LSTM model [4].

From the results in Table 4.11, we notice that BERT is best on eight out of nine datasets except one sarcasm dataset (Reddit), while word2vec CBOW is the second best on four datasets. Overall, our analysis suggests that pre-processing at word embedding stage (Pre) works well for all the three affective tasks.

[28] http://ir.hit.edu.cn/đytang/paper/sswe/embedding-results.zip

[29] https://github.com/facebookresearch/fastText

[30] https://github.com/bfelbo/DeepMoji

[31] https://www.dropbox.com/s/wr5ovupf7yl282x/ewe_uni.txt

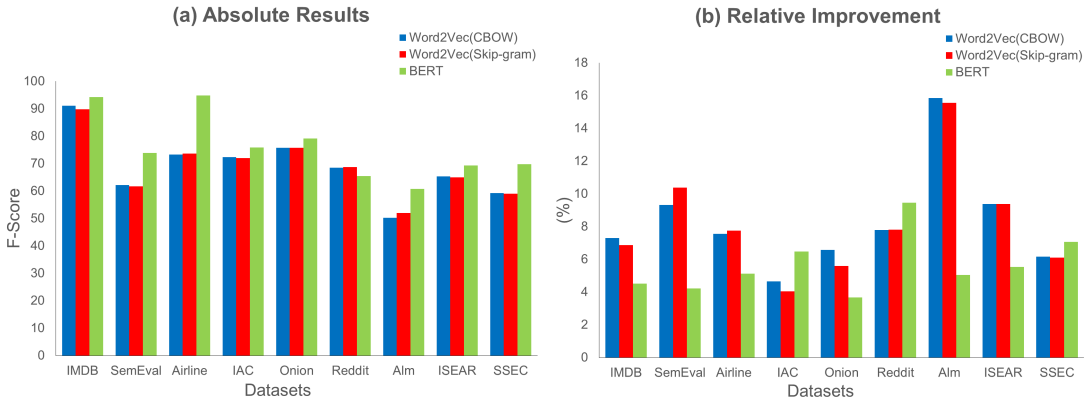

Figure 4.2: Absolute F-scores vs. relative improvement

4.4.4 Analyzing the Three Affective Tasks

Figure 4.2 summarizes the results obtained for all three tasks in terms of (a) absolute F-scores and (b) relative improvement (best pre-processing over `Basic` pre-processing). The IMDB dataset achieves the highest F-score overall, most likely because it consists of movie reviews which are much longer than the text from other genres. As expected, the binary classification task of sentiment analysis and sarcasm detection achieve comparable results, while the multi-class emotion classification typically has much lower F-scores. The most interesting observation, however, is noticed in Fig. 4.2(b) where the emotion datasets show the highest relative improvement, indicating that multi-class classification tasks may benefit the most from applying pre-processing at word embedding stage (Pre).

4.5 Summary

We systematically examined the role of pre-processing training corpora used to induce word representations for affect analysis. While all pre-processing techniques improved performance to a certain extent, our analysis suggests that the most noticeable increase is obtained through negation processing (`neg`). The overall best performance is achieved by applying all the pre-processing techniques, except stop-words removal (All-`stop`). Interestingly, incorporating pre-processing into word representations appears to be far more beneficial than applying it in a downstream task to classification datasets. Moreover, while all the three affective tasks (sentiment analysis, sarcasm detection and emotion classification) benefit from our proposed pre-processing framework, our analysis reveals that the multi-class emotion classification task benefits the most. Exploring the space of subsets of our pre-processing factors might yield more interesting combinations; we leave this for future work.

5 Customized Pre-processing for Word Representation Learning in Affective Tasks

5.1 Introduction

For various natural language processing tasks, word embedding that can capture semantic and syntactic information from contexts have been widely used. However, studies showed existing methods for learning context-based word embeddings typically fail to capture sufficient affective information.

To address the limitations of traditional word embeddings in affective tasks, several techniques such as representing emotion with vectors using a multi-task training framework [196] and generating affective word embeddings [53] have been proposed. One such method [12] using pre-processing in word representation learning in an unsupervised manner was introduced in the previous chapter, with which the results demonstrated that incorporating pre-processing into word representations appears to be far more beneficial than applying it at a downstream task to classification datasets. Regardless of the numer-

ous benefits of incorporating various pre-processing into word representations in affective tasks, one limitation of such approaches is when we apply all the same pre-processing factors on both the bodies of text at training stage and classification datasets appears to be of little benefit (shown in previous chapter Table 4.10). Hence, we hypothesize that the application of different combinations of pre-processing methods in different stages can improve affective tasks results. In particular, our work is motivated by the observation from the previous chapter that while pre-processing factors such as stemming and stop-words removal make up an integral part of nearly every improved affective systems, but they have little effect when they are applied at the training stage on word representation learning.

In the previous chapter, we indicated that applying various pre-processing on word embedding models during training on a certain dataset can improve its performance quality in general and for different affective tasks in particular. Furthermore, not all of the pre-processing methods are considered effective in all affective tasks. For instance, some might even harm the classification results such as stemming and stopwords removal [12] when applied at the embedding-training phase. In addition, others [55, 179] demonstrated that the contribution of use/non-use of stopwords depends on the corpus they applied on for a downstream task, while they showed for some corpora, various pre-processing methods might be even irrelevant. One of the possible explanations for the phenomenon that stopword removal or stemming at training phase harm the performance results in

affective tasks is that not applying these two pre-processing factors may help the word embedding model to better understand the context while applying them at the downstream task can help clean the dataset and reduce the noise.

The aforementioned works describe pre-processing techniques as applied either directly to evaluation datasets in affective systems or examined the effectiveness of directly incorporating these known effective pre-processing techniques further "upstream" into the training corpus of word embeddings, which are widely used across a number of downstream tasks. However, little work has been done towards exploring the influence of different possible combinations of pre-processing techniques applied to word embeddings as well as evaluation datasets based on each affective task. To address this limitation, the overarching goal of this chapter is to perform an extensive and systematic assessment of the effect of a customized combination of linguistic pre-processing factors pre-training to three affective tasks, including *sentiment analysis*, *emotion classification* and *sarcasm detection*.

Towards that end, we systematically analyze the effectiveness of applying customized pre-processing to large training corpora *before* learning word embeddings, and different combination of pre-processing factors when applied on classification datasets which can be more suitable for each affective tasks. Hence, we investigate the following research questions: *(i)* what pre-processing combination(s) is (are) best suited for each affective task?, *(ii)* which combination of pre-processing techniques is more suitable when

they applied on classification tasks and which are more suited for training phase?, *(iii)* does customized pre-processing of word embeddings and downstream tasks provide any improvement over using numerous text pre-processing techniques at once as a general combination for all affective tasks? and if yes, how much?

While the main focus of the previous chapter was studying the role of different pre-processing when they are applied to word embeddings, in this chapter our overreaching goal is to focus on customized combination of pre-processing factors which are more suited for each affective tasks when they are applied on classification dataset. Therefore, We propose a model using various combinations of different pre-processing methods for each affective task individually using different word embeddings models. In brief, the major contributions of this work are as follows:

- We investigate the usefulness of customized pre-processing for word representation learning in affective tasks. In particular, we propose different combination of pre-processing factors that are mostly suitable for each affective task, employing seven embedding models, over nine datasets.

- We study the role of each combination of pre-processing factors for a specific affective task and their major effects on the performance when they are applied in different stages of word embedding in affective analysis.

- We conduct extensive experimental results, showing that, the appropriate combina-

tion of text pre-processing methods for each affective task when applied in different stages of word embedding can significantly enhance the classifier's performance.

- We perform a comparative study of the accuracy performance of word vector models proposed in this work and the one presented in the previous Chapter 4.

The rest of this chapter is organized as follows: Section 5.2 elaborates on the customized pre-processing techniques employed in the evaluation of models for both stages (e.g. training phase and classification datasets) in each affective tasks. Section 5.3 and 5.4 demonstrate the customized pre-processing factors for each affective task. Section 5.5 describes the experimental evaluation framework. In Section 5.6, a comprehensive analysis of the results is provided. Section 5.7 concludes the chapter with key insights of the research.

5.2 Customized Pre-processing in Affective Tasks

This section describes various possible combinations of pre-processing factors that can be applied to the training corpus that is then used to generate word representations and each downstream tasks based on different case studies. Some illustrative examples of text pre-processing for each affective tasks based on case studies are shown in Table 5.1.

Results in the previous chapter demonstrated that a general combination of six pre-processing factors provide an improvement in the performance of three affective tasks

while negation and POS-tagging are the most effective pre-processing factors, when they are applied on the training corpora while training the word embeddings. Moreover, applying the same combination of pre-processing factors on classification datasets or even at both appeared to be of little benefit compared to the embedding-training stage on word embeddings. On the other hand, previous works indicated that using pre-processing factors such as stopword removal and word stemming on classification datasets improve the performance. Even though there is no unique combination of pre-processing tasks that provides successful classification results for every domain, in this chapter our primary goal is to propose a combination that is more suited for each affective task. Below, we present a couple of cases from previous works that demonstrated the role of different combinations of pre-processing factors suited for each affective tasks when they are applied at downstream task on classification datasets.

5.2.1 Sentiment Analysis

Haddi et al., [69] investigated the role of different combination of text pre-processing methods (HTML tags removal, non-alphabetic signs removal, white space removal, pos-tag, abbreviation expansion, stemming, stopwords removal, spell correction, and negation handling) in sentiment analysis of two online datasets of movie reviews, which indicated that reducing noise by text pre-processing improve the performance of the classification task. Jianqiang and Xiaolin [85] indicated that the most effective text pre-processing

methods on sentiment classification are using six pre-processing methods (replacing negative mentions (e.g., "won't" into "will not"), removing URL links, spell correction, removing punctuation, removing stop words, and stemming.

Affective Task	Pre-Processing	Example
Sentiment Analysis	Negation Correction	*I won't walk again.* → *I will not walk again.*
	Negation	*I feel not good.* → *I feel bad.*
	Intensifiers	*This work is extremely hard.*
	Interjections	*Wow, Is this your house?.*
Sarcasm Detection	Emoticons	*Awesome failure!* 😐
	Keeping Punctuation	*Time for your medication or mine ???!!*
	Intensifiers	*Sarcasm detection is too easy!*
	Interjections	*Oh, I'm nicer in sleep.*
Emotion Detection	Emoticons	*Are you serious now??* 😠
	Keeping Punctuation	*We are not friends anymore???!!!*
	Interjections	*Yay! We are going out tonight.*
	Intensifiers	*This is the best experience I've ever had.*

Table 5.1: Example of Case Studies of Different Pre-processing in Affective Tasks.

Furthermore, Carrillo et al. [27] proposed a model of negation, intensifiers, and modality especially conceived for sentiment analysis tasks by handling negation with

an antonym dictionary which replaces the to-be-negated word with its antonym (e.g., "notgood" replaced with "bad"). Hence, based on the experiment in the previous chapters and the aforementioned studies, we plan to study a possible combination of pre-processing factors on classification datasets for the task of sentiment analysis, which includes: negation, pos-tag, stopwords removal except for intensifiers, punctuation removal, spell correction and stemming.

5.2.2 Sarcasm Detection

The results in [25, 59] demonstrated the importance of sarcasm indicators. For instance, Burgers et al. [25] introduced a set of sarcasm indicators that explicitly signal if an utterance is sarcastic, such as emoticons (i.e., ":p", ":)"), spelling correction, interjections (i.e.,"ah", "hmm"), exclamation marks (e.g., "!", "?"), quotation marks, and intensifiers words (e.g., "too", "greatest", "best", "really") that often occur in sarcastic utterances (e.g. "Oh! Sarcasm detection is too easy! :P "). While the importance of pos-tagging stemming [6, 154] and lemmatization [22, 111] was indicated in other studies for sarcasm detection. However, most of these indicators will be removed by applying pre-processing factors such as punctuation removal, stopwords removal and POS tag (i.e. by retrieving only four classes which was suggested in the previous chapter) [12]. Thus, we study a possible combination of pre-processing factors on classification datasets for this task including negation, pos-tag, spell correction, keeping punctuation, emoticons, no stopwords removal,

stemming and lemmatization.

5.2.3 Emotion Detection

In the task of emotion classification, Mulik et al. [129] examined the role of four pre-processing techniques in which the results showed stemming, lemmatization and emoji tagging to be the most effective factors. Moreover, Goddar et al. [61] indicated that interjections such as *yay* and *wow* could be indicators of different emotions, which helps to improve the classification performance. Furthermore, It was observed in [2] that stemming and pos-tag the text does improve the accuracy of the emotion detection process. Whilst, other studies demonstrated the effects of intensifiers, interjections [58, 193], pos-tagging, negation, punctuation [195], and emoticons [193] in emotion detection. Therefore, a possible combination of pre-processing factors on classification datasets for emotion classification includes: negation, pos-tag, spell correction, keeping punctuation, no stopwords removal, emoticons, stemming and lemmatization.

5.2.4 Pre-processing Factors

This section describes the pre-processing factors that we consider to apply to the training corpus that is then used to generate word representations and classification datasets. The details of pre-processing factors including **Basic**, **Spellcheck** (`spell`), **Negation** (`neg`), and **Stemming** (`stem`) is the same as what described in the previous Chapter 4 (Section

§4.2) with the same order. The details of other pre-processing factors are as follows:

Parts-of-Speech (`pos-tag`): Five parts-of-speech classes, namely nouns, verbs, adjectives, adverbs and interjections have been shown to be more informative with regards to affect than the other classes. Thus, using the NLTK pos-tagger, for each sentence in the corpus we retain only the words belonging to all forms of these five classes, i.e., NN*, JJ*, VB*, RB*, UH* .

Custom Stopwords (`c-stop`): Stopwords are generally the most common words in a language typically filtered out before classification tasks. However, as we explained in the previous section, there are benefits for keeping stopwords, in particular since removing them will exclude a number of intensifiers(e.g. too, really) from the text, which literature showed are effective in affective classification tasks. Therefore, in our experiments, we either do not remove the stopwords, or when it's necessary, we use a custom stopwords list excluding intensifier words [32].

Lemmatization (`lemma`): The lemmatization phase is basic to reduce the noise due to the variability or just to have a better understanding of the meaning of a sentence. Lemmatization is the same as stemming in which it changes the word into its root word instead of the stem word. The main difference between lemmatization and stemming is that the lemmatizer considers the morphological analysis of a word (e.g., prefixes,

[32]List of intensifier words are collected from: https://github.com/wyounas/homer and [24]

suffixes and base words). We lemmatize the text using NLTK Wordnet Lemmatizer [33]. In order to find the correct lemma, part-of-speech tagging must be specified. Thus, before lemmatizing, the part of speech tagging must be performed. For example: working, worked, works is lemmatized to "work" in a verb form. When a certain word can belong to more than one grammatical category, depending on its part-of-speech tag within the sentence (e.g., noun or verb), all the related lemmas are kept.

Keeping Punctuation (`k-punc`): Punctuation and special characters such as "@%*=()/ +" are commonly removed from the text. However, studies in [8, 44] indicated the effects of punctuation in affective tasks. Therefore, we keep all the special characters, exclamation marks and punctuation. We make sure there are spaces between words and punctuation during tokenization to keep the punctuation in separate tokens. Moreover, in the case of consecutive punctuation marks (e.g. !!!, :-)), we keep all of them in the same token.

Emoticons (`emojis`): Common habit of using emojis in social media posts and reviews provide evidence of expressing different emotions, and they have been proven [139, 157] to be very effective for various affective tasks. Therefore, we convert graphical emoticons or emojis into text (e.g. converting 😫 to "tired face" and 😏 to "smirking face") using the Demoji module [34]. It is used to accurately remove and replace emojis into text strings.

[33] `https://www.nltk.org/_modules/nltk/stem/wordnet.html`

[34] https://pypi.org/project/demoji/

5.3 Customized Pre-processing Training Corpora

In the previous chapter, results demonstrated that stemming and stopword removal have minimal impact while *negation* and *part-of-speech* resulted as most effective pre-processing factors when they are applied to training corpora. Hence, we conduct our experiments by keeping stopwords and not applying stemming. In contrast with the previous chapter, in this stage we apply customized pre-processing factors including customized pre-processing factors as: `neg`, `pos-tag`, `spell`, `k-punc`, and `emojis`.

5.4 Customized Pre-processing Classification Tasks

This section describes the customized pre-processing factors, which we explained in detail in the previous sections applied to each classification dataset for each affective task.

Sentiment Analysis: `neg`, `pos-tag`, `punc`, `spell`, `c-stop`, and `stem`.

Sarcasm Detection: `neg`, `pos-tag`, `k-punc`, `spell`, `emojis` `lemma`, and `stem`.

Emotion Detection: `neg`, `pos-tag`, `k-punc`, `spell`, `emojis` `lemma`, and `stem`.

5.5 Experimental Evaluation

5.5.1 Training Corpora and Evaluation Datasets

The detail of our training corpora (Wikipedia) is explained in Section §4.3.1 of Chapter 4. In addition, we conduct our evaluation on three tasks, namely sentiment analysis, emotion classification and sarcasm detection. Our evaluation datasets and some illustrative examples of text are explained in Table 4.3 and Table 4.4 of Chapter 4.

5.5.2 Word Embedding Models

We obtain our pre-processed word representations through 4 more models other than the three models (i.e. Word2Vec CBOW, Word2Vec Skip-gram, BERT) we trained in the previous Chapter 4 as follows: (i) **FastText CBOW** and (ii) **FastText Skip-gram**: FastText is another word embedding method that is an extension of the word2vec model [21]. Instead of learning vectors for words directly, FastText represents each word based on sub-word character n-grams. The setting for training this model is also the same as Word2vec for both CBOW and Skipgram model, where n-gram is set to 3. (iii) **Glove**: Global vectors for word representations [142] is a global log-bilinear regression model for the unsupervised learning of word representations, where it creates a global co-occurrence matrix by estimating the probability of a given word that will co-occur with other words. We train the model with the same parameters as the original paper via adaptive gradient

descent (AdaGrad), setting dimensionality to 300, number of epochs to 100 and batch size to 2048. (v) **ELMo**: ELMo (Embeddings from Language Models) is a contextual bidirectional language modeling which is character-based, and the embedding for a given word might vary based on the different contexts in which the word occurs [144]. The hyperparameters for training this model are the same as the original model [144], with LSTM dimensionality 4096, 10 epochs and the number of negative samples batch of 8192.

5.5.3 Classification Setup

For classification, we employ the LSTM model as it works well with sequential data such as text. For binary classification, such as sentiment analysis and sarcasm detection, the loss function used is the binary cross-entropy along with sigmoid activation:

$$\xi = -\frac{1}{N}\sum_{i=1}^{N} y_i log(p(y_i)) + (1 - y_i) log(1 - p(y_i)) \tag{5.1}$$

where y is the binary representation of the true label, $p(y)$ is the predicted probability, and i denotes the i^{th} training sample.

For multiclass emotion classification, the loss function used is categorical cross-entropy loss over a batch of N instances and k classes, along with softmax activation:

$$\xi = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{k} y_{ij} log\left(p(y_{ij})\right) \tag{5.2}$$

where $p(y)$ is the predicted probability distribution, $p(y_{ij}) \in [0, 1]$.

The optimizer is Adam [96], all loss functions are sample-wise, and we take the mean of all samples (epoch = 5, 10, batch size = 64, 128). All sentiment and sarcasm datasets are split into training/testing using 80%/20%, with 10% validation from training. For the smaller and imbalanced emotion datasets, we use stratified 5-fold cross-validation. We use a dropout layer to prevent overfitting by ignoring randomly selected neurons during training. We use early stopping when validation loss stops improving with patience = 3, min-delta = 0.0001. The results are reported in terms of weighted F-score (as some emotion datasets are highly imbalanced), where F-score = $2\frac{p.r}{p+r}$, with p denoting precision, and r is recall.

5.6 Discussion and Analysis

We analyze the impact of customized pre-processing techniques with different combination in word representation learning and classification datasets on affect analysis.

Training Corpus	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
FastText(CBOW)	Basic	74.68	68.20	70.29	57.83	60.50	60.13	48.29	26.07	51.47
	All	77.12	69.86	71.69	60.69	63.39	63.07	50.77	28.42	53.89
	All - `neg`	75.01	68.90	70.83	58.81	61.25	61.74	49.21	27.89	52.04
	All - `pos`	78.51	69.17	70.26	60.57	61.94	62.29	49.86	28.06	52.71
	All - `punc`	76.92	69.37	71.14	60.19	62.71	62.78	50.44	28.30	53.65
	All - `spell`	76.85	69.73	71.00	59.90	62.18	62.41	50.04	28.00	53.65
	All - `stop`	**80.37**	**71.08**	72.39	**62.74**	64.79	64.33	**53.37**	30.24	55.28
	All - `stem`	79.45	70.10	**73.06**	61.83	**65.48**	**65.51**	52.19	**30.61**	**55.74**
FastText(Skip-gram)	Basic	75.00	68.41	70.41	58.13	61.12	60.72	49.13	26.68	52.07
	All	78.30	69.73	71.65	61.52	64.57	63.61	51.03	28.76	54.21
	All - `neg`	75.86	68.59	70.75	59.33	62.03	61.58	50.00	28.04	52.84
	All - `pos`	79.24	70.33	71.00	60.00	63.08	62.11	50.41	29.47	53.07
	All - `punc`	78.01	69.51	71.10	60.94	64.00	62.84	50.94	28.01	53.67
	All - `spell`	77.90	69.50	71.25	61.11	64.27	63.02	50.79	28.63	53.91
	All - `stop`	**81.83**	**71.30**	**73.29**	62.81	**66.12**	65.71	**53.69**	**30.48**	**56.32**
	All - `stem`	80.82	70.82	72.61	**63.28**	65.75	**66.24**	52.76	30.07	56.15

Table 5.2: F-score Results of Evaluating the Effect of Different Pre-processing Factors Using Different Models on Wikipedia Corpus. The Overall Best Results Are in **Bold**.

5.6.1 Effects of General Combination of Pre-processing Factors

Recall that previously in chapter 4, we presented six pre-processing training corpora used to induce word representations for affect analysis and later, we applied the same

pre-processing factors, when applied to classification datasets. In this section, we first conducted the experiments based on the same pre-processing and the same rules as it was explained in the previous chapter on four new word embedding models to identify the most effective pre-processing factors for training word embeddings in affective tasks.

Table 5.2 and 5.3 detail the results of our experiments comparing the performance of ablation studies (i.e., including all the factors but one) on Wikipedia corpus. Observing the performance of the pre-processing factors on the corpus, we note that pre-processing technique can bring improvements on four models when compared to *Basic*, thereby validating our intuition of incorporating pre-processing into training corpora of word representations. Second, negation (`neg`) processing appears to be consistently the most effective factor across all the 9 datasets (same effects as shown in the previous chapter), indicating its importance in affective classification, followed by parts-of-speech (`pos`) processing where we retained words belonging only to one of four classes. On the other hand, removing stopwords (`stop`) and stemming (`stem`) yield little improvement and mixed results, as it was observed in the results of Tables §4.7, §4.8, and §4.9 in the previous chapter. Interestingly, similar to previous chapter results, applying all the pre-processing factors is barely better except for the GloVe model. A possible explanation for these observations can be related to the basic idea behind the GloVe model, which is to derive the relationship between the words from *Global Statistics*.

Training Corpus	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
GloVe	Basic	83.51	69.12	70.01	69.48	70.21	62.76	54.31	64.77	56.33
	All	**87.32**	**73.22**	**74.39**	**73.14**	**74.19**	**67.29**	58.51	**67.34**	**59.70**
	All - `neg`	84.06	70.09	71.37	71.15	71.39	63.24	55.10	65.31	57.63
	All - `pos`	85.33	72.76	72.19	72.35	72.06	65.07	56.33	62.82	58.12
	All - `punc`	86.72	71.47	72.77	72.62	73.95	65.88	57.90	66.74	59.37
	All - `spell`	86.48	71.28	73.61	72.69	73.70	65.79	58.04	66.70	58.42
	All - `stop`	86.39	72.84	73.64	72.84	73.67	66.19	57.61	67.13	59.10
	All - `stem`	86.25	73.00	73.41	72.33	73.61	66.25	**58.93**	66.05	59.33
ELMo	Basic	86.34	69.47	82.11	70.18	71.65	65.48	60.24	65.00	66.20
	All	88.63	71.80	83.91	72.61	73.30	66.93	63.20	67.58	69.45
	All - `neg`	87.00	70.21	82.79	71.58	72.00	66.10	61.06	66.10	67.24
	All - `pos`	87.64	70.68	83.00	72.00	72.49	66.47	61.70	65.72	67.81
	All - `punc`	88.41	71.67	83.27	72.40	73.17	66.71	62.47	67.00	68.73
	All - `spell`	88.23	71.55	83.16	72.33	73.09	66.50	62.59	67.11	68.10
	All - `stop`	**90.27**	**73.54**	**85.61**	**74.04**	**74.45**	**68.74**	64.17	**69.28**	**71.01**
	All - `stem`	89.45	72.30	85.02	73.10	74.20	67.59	**64.78**	68.52	70.33

Table 5.3: F-score Results of Evaluating the Effect of Different Pre-processing Factors Using Different Models on Wikipedia Corpus. The Overall Best Results Are in **Bold**.

The model looks at the co-occurrence matrix that tells us how often a particular pair of words occur together. Each value in a co-occurrence matrix is a count of a pair of words occurring together.

Moreover, in Table 5.2, we note that the best performance for the FastText (CBOW)

model in majority comes from combining all the pre-processing factors except stemming (All-`stem`), whereas for the FastText (Skip-gram) and ELMo models, the best results are obtained by applying all the pre-processing factors except stopwords removal (All-`stop`). In addition, we need to note that FastText works well with rare words. Thus, even if a word wasn't seen during training, it can be broken down into n-grams to get its embeddings. However, ELMo is character-based, which means that the model does not have a pre-defined vocabulary of words used for training, but it rather extracts the word embeddings from the constituent characters of the words. These differences can make the word embedder more robust and generalizable to downstream tasks.

In Table 5.4, we investigate the difference between applying the general combination of pre-processing to the training corpora for generating word embeddings (Pre) and applying the same pre-processing to the classification datasets (Post). As an example, during *Pre*, we first apply the pre-processing techniques (e.g., all but stemming) to the training corpus (e.g., Wikipedia), then generate word embeddings, further convert a classification dataset (e.g., IMDB) into word embedding representation, and finally classify using LSTM. Conversely, for *Post*, we first generate word embeddings from a training corpus (e.g., Wikipedia), then apply the pre-processing techniques (e.g., all but stemming) to the classification dataset (e.g., IMDB), which is then converted to word vector representation, and finally classified using LSTM [35].

[35]Note: For settings including stemming, the classification data is also stemmed in order to obtain a

Training Corpus	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
FastText(CBOW)	Pre	79.45	70.10	**73.06**	**61.83**	**65.48**	**65.51**	52.19	30.61	**55.74**
	Post	76.48	69.25	70.15	57.38	62.48	63.71	51.47	29.35	53.84
	Both	**79.72**	**70.54**	72.26	61.00	65.27	65.20	**52.36**	**30.65**	55.49
FastText(Skip-gram)	Pre	**81.83**	**71.30**	**73.29**	62.81	66.12	**65.71**	**53.69**	**30.48**	**56.32**
	Post	80.01	70.16	71.40	58.70	64.22	63.76	51.49	29.74	54.38
	Both	80.52	70.40	72.58	**63.02**	**66.57**	65.00	53.18	30.24	55.29
GloVe	Pre	**87.32**	**73.22**	**74.39**	**73.14**	**74.19**	67.29	58.51	**67.34**	**59.70**
	Post	86.37	71.20	72.30	72.15	72.47	65.73	57.19	66.31	58.64
	Both	87.00	72.48	74.18	73.01	73.61	**67.32**	**58.66**	67.29	59.14
ELMo	Pre	**90.27**	**73.54**	**85.61**	**74.04**	**74.45**	**68.74**	64.17	**69.28**	**71.01**
	Post	88.13	71.76	83.10	72.28	73.55	66.79	63.80	68.20	70.18
	Both	90.14	72.57	85.00	73.61	74.20	68.07	**64.39**	68.79	70.83

Table 5.4: F-score Results of Evaluating the Effect of Pre-processing Word Embeddings Training Corpus vs. Pre-processing Evaluation Datasets

The results of this experiment are presented in Table 5.4, where similar to previous chapter results, we observe that incorporating pre-processing into the training corpora before generating word vectors (Pre) outperforms pre-processing classification datasets (Post) across all nine datasets of the three affective tasks. Similarly though, we observe pre-processing both the bodies of text (Both) appears to be of little benefit, thereby

compatible vocabulary.

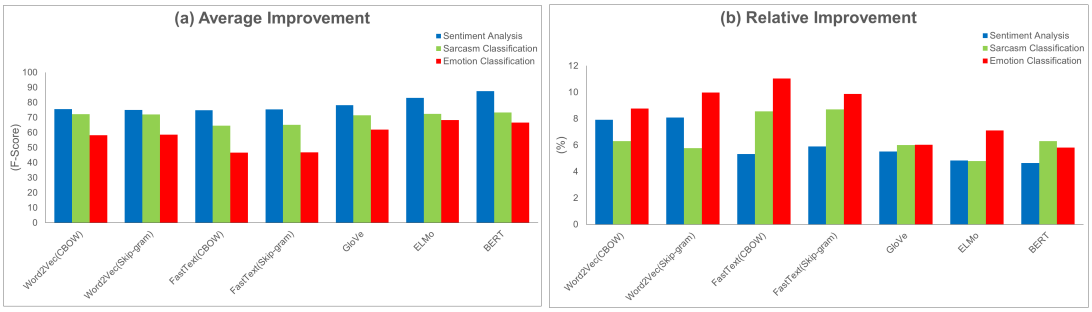

Figure 5.1: Average F-scores vs. relative improvement

validating our intuition of customized combination of pre-processing at downstream task might result in better performance.

5.6.2 Analyzing the Three Affective Tasks

Figure 5.1 summarizes the results obtained from all the seven word embedding models (three models from the previous chapter and 4 models from this chapter) for all three tasks in terms of (a) absolute F-scores and (b) relative improvement (best pre-processing over `Basic` pre-processing). The IMDB dataset achieves the highest F-score overall, most likely because it consists of movie reviews, which are much longer than the text from other genres. As expected, the binary classification task of sentiment analysis and sarcasm detection achieve comparable results, while the multi-class emotion classification typically has much lower F-scores. The most interesting observation, however, is noticed in Fig. 5.1(b), where the emotion datasets show the highest relative improvement, indicating that multi-class classification tasks may benefit the most from applying pre-processing at

word embedding stage (Pre).

In particular, Figure 5.1 summarizes the results obtained for all three tasks, where we averaged all three datasets in each task from the best results in Table 5.2 and 5.3 in bold. The sentiment analysis datasets (blue bar) achieves the highest F-score overall, most likely because it consists of movie reviews which are much longer than the text from other genres. As expected, the binary classification task of sentiment analysis and sarcasm detection achieve comparable results, while the multi-class emotion classification typically has much lower F-scores. The most interesting observation, however, is noticed in Fig. 5.1(b) relative improvement of the best results for each model over "Basic" for Wikipedia where the emotion datasets (red bar) show the highest relative improvement, indicating that multi-class classification tasks may benefit the most from applying pre-processing at word embedding stage (Pre).

5.6.3 Effects of Customized Pre-processing Factors for Each Affective Task

A primary goal of this chapter is to identify the effects of customized pre-processing factors for training word embeddings for affective tasks as well as evaluation datasets.

Training Corpus	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
Word2Vec(CBOW)	All	88.41	60.25	71.39	71.57	73.61	65.27	48.81	62.48	57.42
	All - `stem`	88.76	62.19	73.25	72.36	75.69	68.53	50.28	65.33	59.28
	`c-pre`	**90.67**	**62.74**	**74.33**	**73.08**	**76.52**	**69.15**	**53.18**	**66.19**	**60.51**
Word2Vec(Skip-gram)	All	87.00	59.99	71.29	71.25	73.82	65.67	48.51	**65.02**	57.13
	All - `stop`	89.76	61.74	72.19	72.00	75.69	68.29	52.01	64.00	58.14
	`c-pre`	**89.91**	**62.73**	**73.69**	**72.85**	**76.31**	**69.24**	**52.84**	64.80	**59.28**
FastText(CBOW)	All	77.12	69.86	71.69	60.69	63.39	63.07	50.77	28.42	53.89
	All - `stem`	79.45	70.10	73.06	61.83	65.48	65.51	52.19	30.61	55.74
	`c-pre`	**80.71**	**71.90**	**73.70**	**63.17**	**66.24**	**66.71**	**53.00**	**33.25**	**56.49**
FastText(Skip-gram)	All	78.30	69.73	71.65	61.52	64.57	63.61	51.03	28.76	54.21
	All - `stop`	81.83	71.30	73.29	62.81	66.12	65.71	53.69	30.48	56.32
	`c-pre`	**82.93**	**72.00**	**74.15**	**63.57**	**66.80**	**66.79**	**55.38**	**32.29**	**56.63**
GloVe	All	**87.32**	73.22	**74.39**	73.14	74.19	67.29	58.51	**67.34**	59.70
	`c-pre`	86.73	**73.41**	74.00	**74.23**	**74.27**	**68.40**	**59.80**	66.85	**60.11**
ELMo	All	88.63	71.80	83.91	72.61	73.30	66.93	63.20	67.58	69.45
	All - `stop`	90.27	**73.54**	**85.61**	70.04	74.45	68.74	64.17	69.28	71.01
	`c-pre`	**90.40**	73.20	85.03	**71.19**	**75.27**	**69.87**	**65.38**	**69.81**	**71.80**
BERT	All	91.86	71.76	91.73	73.66	78.72	62.60	59.74	67.80	67.49
	All - `stop`	**94.18**	73.81	94.85	78.80	79.10	65.39	60.73	69.33	69.81
	`c-pre`	93.67	**74.00**	**94.88**	**79.00**	**79.84**	**66.00**	**61.18**	**70.28**	**70.33**

Table 5.5: F-score Results of Comparing the Effect of Customized Pre-processing Vs General Pre-processing

Table 5.5 details the results of our experiments comparing the performance of applying all pre-processing (`All`) on Wikipedia corpus and the best results of ablation studies (e.g. `All-stem` from the previous chapter with customized pre-processing (e.g. `c-pre` for all the seven models. Observing the performance of the customized pre-processing factors (`c-pre`) when we apply it on the training corpus, we note that customized pre-processing techniques constantly in most word embedding models outperform the general combination of pre-processing (e.g. `All`, or `All-stem`), thereby validating our intuition of incorporating customized pre-processing into training corpora of word representations perform better in most cases than those in the previous chapter.

5.6.4 Evaluating Pre-processing Training Corpora vs. Pre-processing Classification Dataset

In this section, we compare the performance of customized pre-processing applied to the training corpora for generating word embeddings (`c-pre`) against applying them in four different ways: (i) Post 1: applying the same pre-processing applied to training corpora at classification datasets, (ii) Both 1: applying the same customized pre-processing at both stages, (iii) Post 2: applying customized pre-processing based on each affective task on classification datasets (which was explained in section 5.4), and Both 2: applying customized pre-processing factors for training corpus and customized pre-processing for each affective task at classification datasets. As an example, during (`c-pre`), we first

apply the customized pre-processing techniques explained in Section 5.3 to the training corpus (e.g., Wikipedia), then generate word embeddings, then convert a classification dataset (e.g., IMDB) into word embedding representation, and finally classify using LSTM Conversely, for *Post 1*, we first generate word embeddings (using Basic pre-processing) from a training corpus (e.g., Wikipedia), then apply the same customized pre-processing techniques used in `c-pre` to the classification dataset (e.g., IMDB), which is then converted to word vector representation, and finally classified using LSTM. While in *Post 2*, we apply different variations of customized pre-processing methods for each affective task, which was described in Section 5.4.

The results of these experiments are presented in Table 5.4, where we observe that incorporating customized pre-processing into the training corpora before generating word vectors (`c-pre`) outperforms pre-processing classification datasets (Post 1) and (Post 2) in the majority of the cases across all nine datasets of the three affective tasks while applying at (Both 1) still harm the performance in most cases. Interestingly, this time customized pre-processing both the bodies of text (Both 2) appears to be more beneficial compared to apply them at single stage individually, suggesting that an appropriate combination of customized pre-processing at each stage achieves better results than applying them only on one stage or using the general pre-processing factors for affective tasks.

Training Corpus	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
Word2Vec(CBOW)	c-pre	90.67	62.74	74.33	73.08	76.52	69.15	53.18	66.19	60.51
	Post 1	87.30	60.04	72.20	68.27	73.61	66.80	48.25	61.29	55.00
	Both 1	88.13	61.70	72.69	70.08	74.12	68.48	50.23	65.37	58.00
	Post 2	88.52	60.47	73.40	71.25	75.63	68.05	50.44	64.20	59.31
	Both 2	**90.81**	**63.30**	**75.07**	**74.69**	**77.80**	**70.51**	**54.20**	**67.48**	**61.02**
Word2Vec(Skip-gram)	c-pre	89.91	62.73	73.69	72.85	76.31	69.24	52.84	64.80	59.28
	Post 1	88.01	60.22	70.25	71.13	74.28	67.45	50.62	62.00	55.70
	Both 1	88.57	61.85	73.20	71.08	75.00	69.00	50.74	63.12	57.21
	Post 2	89.10	62.03	72.45	71.62	75.69	68.14	51.66	62.70	58.00
	Both 2	**90.40**	**64.20**	**75.37**	**74.28**	**77.82**	**71.49**	**54.09**	**66.00**	**60.58**
BERT	c-pre	93.67	74.00	94.88	79.00	79.84	66.00	61.18	70.28	70.33
	Post 1	91.83	70.12	92.00	74.04	76.81	62.71	58.02	67.90	66.80
	Both 1	94.03	72.19	92.20	76.39	77.19	63.77	60.03	68.34	67.61
	Post 2	93.10	73.24	92.60	75.00	78.20	64.80	59.34	69.18	69.52
	Both 2	**94.22**	**75.20**	**94.88**	**80.21**	**80.34**	**67.41**	**63.10**	**72.66**	**72.80**

Table 5.6: F-score Results of Evaluating the Effect of Customized Pre-processing Word Embeddings Training Corpus vs. Customized Pre-processing Evaluation Datasets

Training Corpus	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
FastText(CBOW)	c-pre	80.71	71.90	73.70	63.17	66.24	66.71	53.00	33.25	56.49
	Post 1	77.30	70.10	71.27	56.80	65.30	63.78	52.67	30.27	54.18
	Both 1	78.69	71.25	71.69	61.38	65.84	64.37	52.73	32.80	54.80
	Post 2	78.29	70.18	71.02	60.39	66.18	65.01	53.00	33.28	55.70
	Both 2	**81.60**	**72.50**	**75.06**	**65.86**	**68.21**	**69.17**	**55.48**	**36.45**	**58.71**
FastText(Skip-gram)	c-pre	82.93	72.00	74.15	63.57	66.80	66.79	55.38	32.29	56.63
	Post 1	78.20	67.84	70.33	59.67	63.80	61.30	51.27	30.69	54.70
	Both 1	79.06	70.60	73.12	62.81	65.30	64.80	54.25	30.39	55.00
	Post 2	80.83	69.38	72.65	62.30	65.30	64.27	55.18	31.40	55.80
	Both 2	**83.60**	**73.41**	**75.33**	**65.39**	**68.42**	**68.70**	**57.04**	**35.20**	**58.00**
GloVe	c-pre	86.73	73.41	74.00	74.23	74.27	68.40	59.80	66.85	60.11
	Post 1	85.12	70.00	71.45	72.64	71.69	65.10	56.48	64.23	57.29
	Both 1	87.29	73.00	73.14	73.45	73.59	67.48	58.29	66.70	58.76
	Post 2	86.20	72.00	73.10	73.00	73.81	66.80	58.30	65.10	58.26
	Both 2	**87.23**	**75.08**	**75.14**	**74.40**	**76.31**	**70.25**	**61.40**	**68.71**	**62.30**
ELMo	c-pre	90.40	73.20	85.03	71.19	75.27	69.87	65.38	69.81	71.80
	Post 1	86.25	70.33	82.20	69.48	73.02	66.40	63.14	67.40	69.28
	Both 1	90.33	72.60	83.20	72.68	74.11	68.00	64.21	67.62	70.37
	Post 2	88.67	72.80	84.61	70.39	74.69	68.80	64.20	68.07	70.30
	Both 2	**91.20**	**74.83**	**86.67**	**73.30**	**77.00**	**71.37**	**67.49**	**71.25**	**72.20**

Table 5.7: F-score Results of Evaluating the Effect of Customized Pre-processing Word Embeddings Training Corpus vs. Customized Pre-processing Evaluation Datasets

5.7 Summary

In this chapter, we systematically examined the role of customized pre-processing training corpora as well as classification datasets for affect analysis. Our analysis reveals that appropriate combination of pre-processing on training corpora and evaluation dataset better improved the performance rather than the general combination of pre-processing which was presented in the previous chapter. The overall best performance between all word embedding models is achieved by BERT by applying all the customized pre-processing techniques at both stages over nine datasets. Interestingly, this time incorporating customized pre-processing at `Both 2` training corpora and the downstream task appears to be far more beneficial than applying them individually or applying the same factors at both stages. Moreover, while all the three affective tasks (sentiment analysis, sarcasm detection and emotion classification) benefit from our proposed customized pre-processing framework, our analysis reveals that the multi-class emotion classification task benefits the most.

6 Affective and Contextual Embedding for Affect Detection

6.1 Introduction

The most popular affective task is known as sentiment analysis, which is the interpretation and classification of emotions (positive, negative and neutral) within text data. Whilst, emotion detection entails classifying the text into fine-grained categories of emotions such as happiness, sadness, fear, and so on. The task of identifying emotions in text is especially difficult due to limited resources, lack of linguistic information, and because it requires a larger number of categories of emotions in which to undertake classification. Another affect analysis is the recognition of sarcasm, which is a particular case of emotion analysis where the emphasis is on sarcasm instead of identifying a sentiment across the continuum. The task of this field is therefore to detect whether or not a given text is sarcastic. Unlike in the study of sentiments where the types of sentiments (e.g. positive, negative,..) are very well defined (e.g. "love" has a positive feeling logically, while "hate"

has a negative feeling, no matter whom you ask or what language you speak), the borders of sarcasm are not so well defined. Moreover, it is important to have a notion of what sarcasm is before beginning to detect it.

Sarcasm is the use of language in which one conveys implicit information/intention with the opposite meaning of what is said or written. Due to this deliberate ambiguity, sarcasm detection is a more challenging task than other affective tasks (e.g. sentiment analysis, emotion detection), especially in written expressions where *body gestures*, *tone of voice*, and *facial expression* are not known [86, 163]. Sarcasm detection has attracted growing interest over the past decade as it draws a more accurate picture of users' intention on social media [28], and facilitates accurate sentiment analysis in online comments and reviews [56]. It also has useful applications in areas such as healthcare [32], hate speech detection [46, 128], disaster management [56].

Early attempts of sarcasm detection from text mainly relied on extracting a set of positive verbs and negative/undesirable situations (e.g. "I love [positive verb] the pain of breakup [negative situation]") [63, 154]. Alternatively, one may use lexical features (e.g., capital letters and excessive usage of exclamatory marks) [108] in sarcasm detection. Recently, psychological studies have shown a strong relationship between affect/sentiment features (e.g., sadness, happiness) and sarcasm [83, 147].

However, relying only on affect/sentiment features for sarcasm detection may not be effective, especially when there are no sentiment words in a sentence [87, 154]. For

instance, in the sentence “Is it time for your medication or mine?”, the speaker’s intention is to mock the person addressed, but there aren’t any sentiment words used.

Later attempts for sarcasm detection mostly relied on language models that are based on continuous representation or embeddings of words, such as Word2vec [116] and GloVe [142]. The use of these general models can eliminate the need for feature engineering or dependence on enormous emotion labeled datasets. However, due to the mechanism with which word vectors are learned and embedded into a space, these models have been shown to be inadequate for affective tasks [10]. For instance, two dissimilar words like “good” and “bad”, which often occur in a similar context, will be embedded closer to each other than the words “good” and “happy” that express similar emotions. More recent advances in neural representation models, such as ELMo [143] and BERT [45] can overcome this limitation by taking into account both the context a word appears in and its importance in that context, using a self-attention mechanism [183]. While these models have been applied successfully to a wide range of Natural Language Processing (NLP) tasks, such as sentiment analysis [169] and word similarity computation [205], they have not been fully exploited for the more challenging problem of sarcasm detection. A few studies that proposed to use these models [30, 128], utilize the already pre-trained embeddings, which are not optimized for sarcasm detection, and their performance can be further improved [149].

In this chapter, we propose two novel models for sarcasm detection based on Affective

and Contextual Embeddings, namely **ACE 1** and **ACE 2**. Given as input a text passage (i.e., a sequence of sentences, which we call a *document* in this chapter for brevity), the models predict whether it is sarcastic. In addition, we evaluate the performance of the proposed models on other affective tasks such as emotion detection and sentiment analysis. The architecture of each model builds upon two components: a) *affective feature embedding*, and b) *contextual feature embedding*. The former utilizes a Bi-LSTM with multi-head attention neural architecture [183] to obtain representations of *affective features* of a document. The latter is achieved by a BERT model. In ACE 1, the two components are combined by training a new BERT model from scratch by adding affective feature embeddings into the input sequence of BERT so that task-specific embeddings can be obtained. In ACE 2, the two components are combined in a fully connected layer with a softmax to form a classifier that is trained with labeled sarcasm detection data. The main contributions of this work are as follows:

- We present two novel deep neural network language models (ACE 1 and ACE 2) for sarcasm detection. Each model extends the architecture of BERT by incorporating both affective and contextual features of text to build a classifier that can determine whether a document is sarcastic or not. To the best of our knowledge, this is the first attempt to directly alter BERT's architecture (rather than using the already pre-trained BERT embeddings) and train it from scratch to build a sarcasm classifier.

- Integral to our proposed models is a novel model that learns the affective representation of a document, using a Bi-LSTM architecture with multi-head attention. The resulting representation takes into account the importance of the affect representations of the sentences in the document.

- We design and evaluate alternatives that materialize each of the two components (affective feature embedding and contextual feature embedding) of the proposed deep neural network architecture model. We systematically evaluate the effectiveness of each alternative architecture.

- We conduct an extensive evaluation of the performance of the proposed models (ACE 1 and ACE 2), which demonstrates that they significantly outperform current state-of-the-art models for sarcasm detection.

- We investigate whether the proposed affective and contextual model (ACE 1 and ACE 2), can improve the performance of other tasks, such as emotion detection and sentiment analysis. Furthermore, we form a comparative study of the accuracy performance of previous BERT model using pre-processing for affective tasks with the current proposed models.

- We make *source code* and *data* publicly available to encourage result reproducibility and model re-use[36].

[36] `https://github.com/NastaranBa/ACE-for-Sarcasm-Detection`

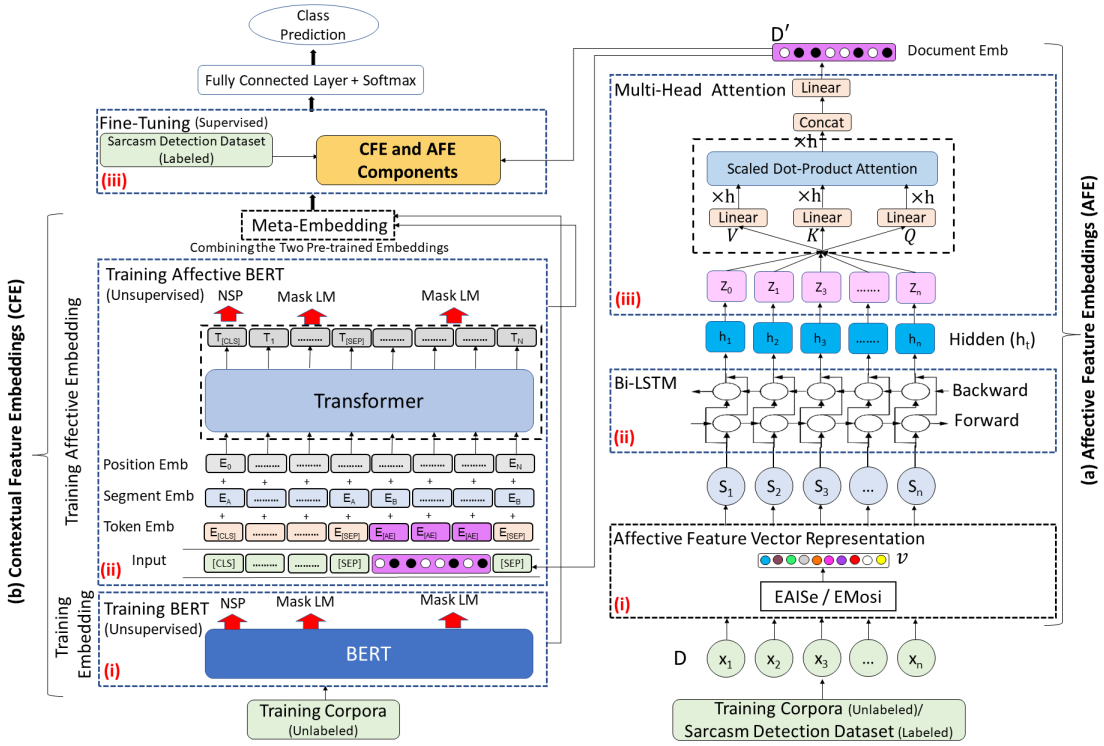

Figure 6.1: Overview of the proposed model ACE 1

The chapter is organized as follows. Section 2 presents the proposed models for sarcasm detection. Section 3 describes the experimental evaluation and Section 4 demonstrates the results and analysis. We conclude the chapter in Section 6.

6.2 Proposed Models for Sarcasm Detection

We propose two models, ACE 1 and ACE 2, for sarcasm detection, where each model takes a document (i.e., a sequence of sentences) as input and predicts whether the document is sarcastic or not. Figure 6.1 and §6.2 depict the architecture of each model that builds upon

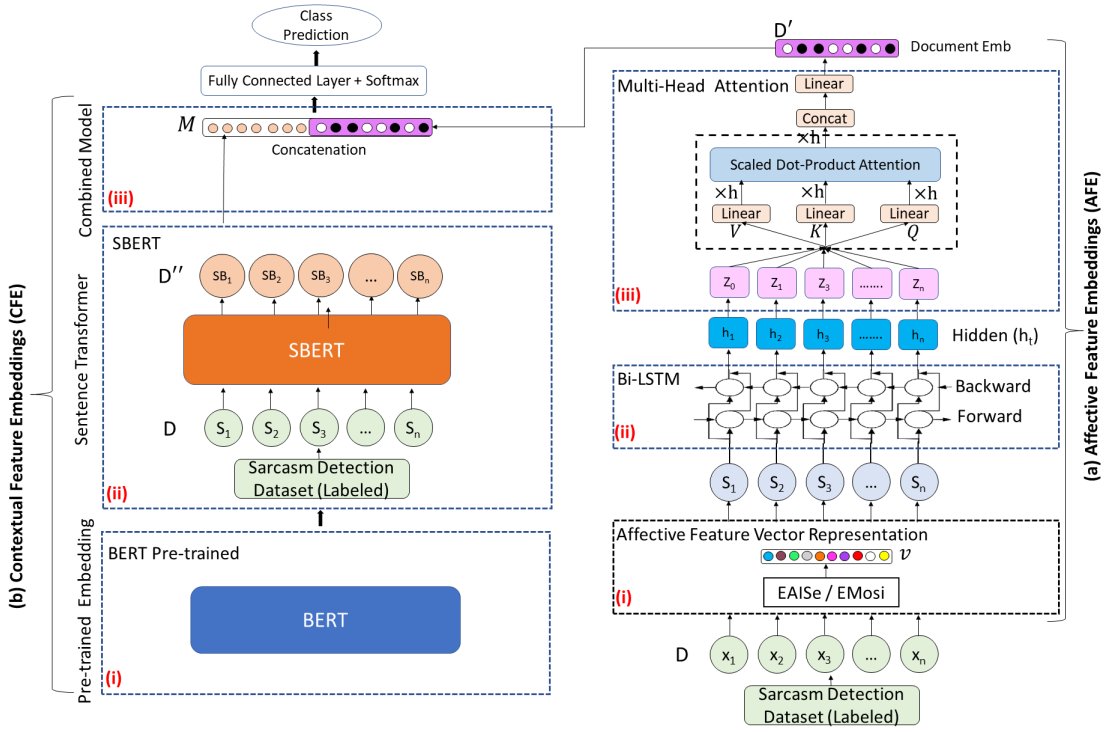

Figure 6.2: Overview of the proposed model ACE 2

two components: a) *affective feature embedding (AFE)* (on the right), and b) *contextual feature embedding (CFE)* (on the left). The two models are different in (1) the way the two components are combined and (2) the input to the affective feature embedding component.

6.2.1 Affective Feature Embedding (AFE)

The architecture of the AFE component is the same for ACE 1 and ACE 2. The difference is that its input in ACE 1 is an unlabeled training corpus in pre-training and a labeled sarcasm dataset during fine-tuning, while in ACE 2, AFE only takes the labeled sarcasm

detection dataset as the input. This component includes three stages: (i) Affective Feature Vector Representation, (ii) Bi-LSTM and (iii) Multi-Head Attention layers.

6.2.1.1 Affective Feature Vector Representation

In this stage, each input document is first chunked into sentences. Then, the affective features are extracted using one of the following two approaches.

Emotion Affective Intensity with Sentiment Feature (EAISe): We use the NRC Emotion Intensity Lexicon [121] to extract the emotion words in a sentence and give each such word 4 intensity scores, one for each of 4 basic emotions: anger, fear, sadness, joy. Each score ranges from 0 to 1, where "1" means that the word conveys the highest degree of the corresponding emotion, and "0" means that the word is not associated with the emotion. Then, we add 2 more binary scores to represent the sentiment (positive, negative) of the word based on the NRC Emotion Lexicon [124]. To calculate the *affective feature vector of a sentence*, we first average the affective feature vectors of the affect words in the sentence, then multiply it with a vector v' that contains the frequency of words in the sentence in each emotion or the sentiment (See Figure 6.3). For instance, assume that we have 3 affect words in a sentence, and the affective feature vectors of 4 emotions and 2 sentiments (anger, fear, sadness, joy, positive, negative) for these words are: $w_{tragedy} = (0, 0.73, 0.61, 0, 0, 1)$, $w_{thanksgiving} = (0, 0, 0, 0.64, 1, 0)$, $w_{happy} = (0, 0, 0, 0.82, 1, 0)$. The element-wise multiplication of the average of these

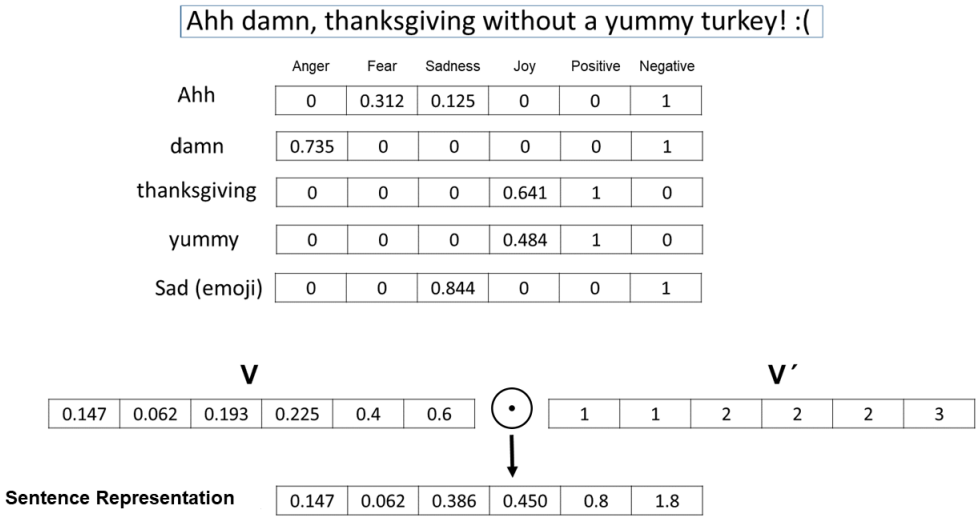

Figure 6.3: Overview of the proposed model ACE 2

3 vectors $(0, 0.24, 0.20, 0.48, 0.66, 0.33)$ and the frequency vector $v\prime = (0, 1, 1, 2, 2, 1)$ is $(0, 0.24, 0.20, 0.96, 1.32, 0.33)$.

Emotion Similarity Feature (EMoSi): In this approach, for each word in a sentence, we measure the average semantic similarity score between the words in the sentence and all seed words (20 words) of an emotion in the NRC Emotion Lexicon [124] that has 8 emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust). The semantic similarity score is based on the cosine similarity of pre-trained Word2vec vectors [115] of the corresponding words. For instance, in a sentence "I love jogging.", we calculate the cosine similarity between word "I" with a seed word "happy" in emotion "joy". We do this for all the seed words in each emotion, resulting in an emotion intensity vector of 8

scores for each word in the sentence. By averaging the vectors for all the words in the sentence, we obtain an affective feature representation of the sentence.

Given a document D with n sentences $(x_1, x_2, \cdots, x_n)$, x_i is converted into its affective vector representation S_i using one of the above two approaches. Thus, document D can be represented as $(S_1, S_2, .., S_n)$.

6.2.1.2 Bi-LSTM Layer

Given a document $D = (S_1, S_2, .., S_n)$, we use a Bi-LSTM model [66] to capture/encode the affect-changing information of the sentence sequence from both left and right directions[37]. The result is a sequence of hidden state vectors h_t for the sentence sequence. More precisely, the bidirectional LSTM is a concatenation of the forward and backward LSTM as:

$$\overrightarrow{h}_t = \overrightarrow{LSTM}(S_t, \overrightarrow{h}_{t-1}) \tag{6.1}$$

$$\overleftarrow{h}_t = \overleftarrow{LSTM}(S_t, \overleftarrow{h}_{t+1}) \tag{6.2}$$

$$h_t = Concat(\overrightarrow{h}_t, \overleftarrow{h}_t) \tag{6.3}$$

The sequence of hidden state vectors h_t $(t = 1, 2, \ldots n)$ forms a matrix and serves as the input for the next layer (Multi-head Attention Layer).

[37]We use post-zero-padding , and using attention masking to distinguish the padding[183]

6.2.1.3 Multi-head Attention Layer

In a given document, a specific part could play a more important role in detecting sarcasm [99]. Therefore, we use a multiple heads attention mechanism [183] to capture the importance of hidden affective feature vectors h_t, which have already been learned by the Bi-LSTM layer. This helps us capture long-distance dependencies and form a global representation of the sequence. As shown in Figure6.1, the output vectors of Bi-LSTM layer $(h_1, h_2, \ldots h_n)$ are combined to form a matrix $H = [h_1, h_2, ..h_n]$ which serves as the input matrix for each attention head in the *self-attention* mechanism. In particular, the three matrices Q(query), K(key), and V(value) are created by multiplying H with the weight matrices (W_Q, W_K, W_V) that are trained jointly in the *self-attention* mechanism [183] as such: $Q = H \times W^Q$, $K = H \times W^K$, and $V = H \times W^V$. Then, this mechanism calculates the context vectors for each self-attention head as:

$$Z = SDPA(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_K}}\right)V \tag{6.4}$$

This is called Scaled Dot-Product Attention (SDPA) in [183] and $\sqrt{d_K}$ is a scaling factor where d_K is the dimension of queries and keys. Assuming there are N heads, we have N such sets of weight matrices and we compute the output for the $i'th$ head as:

$$Z(head_i) = SDPA(QW_i^Q, KW_i{}^K, VW_i^V) \tag{6.5}$$

The Multi-Head Attention mechanism (MHA) runs through SDPA multiple times in parallel and concatenates the resulting vectors of all the heads and multiply it by an additional

weight matrix W^O that was trained jointly with the model as the final representation of the document D as:

$$D' = MHA(Q, K, V) = Concat(Z_1(head_1), ..., Z_n(head_N)W^O \tag{6.6}$$

6.2.2 Contextual Feature Embedding in ACE 1

We explain how our first model, ACE 1, incorporates the affective feature embedding discussed in Section 6.2.1 into the BERT model for sarcasm detection. The architecture of the ACE 1 is illustrated in Figure 6.1, which includes three stages: i) Training BERT, ii) Training Affective BERT and iii) Fine-Tuning.

6.2.2.1 Training BERT

In this stage, we train a BERT model using the BERT-Large-uncased architecture with the same setting for hyper-parameters as in [45], where more details on used parameters can be found in section §6.3.3. The model is trained on an unlabeled text corpus over two unsupervised tasks: *i*) *Masked Language Model* (MLM), in which, some of the tokens in the input sequences are masked and the model is trained to predict these masked tokens, and *ii*) *Next sentence Prediction* (NSP), where the model receives pairs of sentences as input and learns to predict if the second sentence in a pair is the subsequent sentence of the first one in the training corpus. The two tasks are trained together, with the goal of minimizing the combined loss function to generate the final embedding vectors.

More specifically, the training corpus is tokenized with the WordPiece method [190]) and then input sequences are generated, where 50% are a pair in which the second sentence is the subsequent sentence in the corpus, and the other 50% contains a random sentence from the corpus as the second sentence. Each input sequence has a [CLS] token at the beginning and a [SEP] token at the end of each sentence. The middle part of Figure 6.4 illustrates the BERT input representation. BERT has three embedding layers: (i) Token Embedding: it transforms tokens into vector representations of fixed dimension from the WordPiece token vocabulary (ii) Segment Embedding: it discerns between the first and second sequence to indicate whether the token belongs to the first or the second sentence in the input sequence, and (iii) Position Embedding: it remembers the position of each token in a sequence. These three embeddings are summed up (element-wise) and make up the input to the BERT bidirectional transformer which is a multi-layer bidirectional transformer encoder [45, 183].

6.2.2.2 Training Affective BERT with Affective Feature Embeddings

The purpose of this stage is to incorporate the affective features into the BERT model so that task-specific embeddings can be obtained. As illustrated in Figure 6.1, we train a new model (called Affective BERT) from scratch using BERT by adding affective feature embeddings obtained from the AFE component into the input sequence of BERT. Again, we use the BERT-Large-uncased architecture. The unlabeled training corpus is

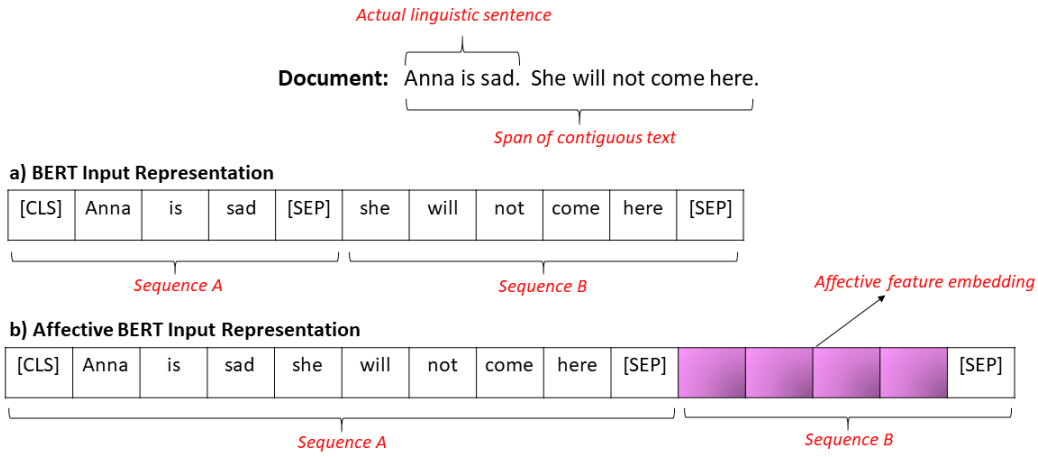

Figure 6.4: BERT Input Representation Vs Affective BERT Input Representation.

first tokenized using the WordPiece method. The difference between this BERT model and the one trained in the first stage is in the input sequence. The bottom part of Figure 6.4 illustrates an input sequence in this stage. An input sequence contains two subsequences. The first one (between [CLS] and the first [SEP]) is a document (i.e., a sequence of tokens) and the second subsequence (purpule cells between two [SEP] tokens) is the affective feature embedding of the document, which is the D' vector generated by the AFE component trained earlier using the training corpus. The BERT model is trained using two tasks as usual: *Masked Language Model* (MLM) and *Next Sentence Prediction* (NSP). Since our input sequence contains a document and its affective feature embedding, the NSP task is actually to predict the affective features of a document.

Now, we have two contextual pre-trained embeddings for each token: A from the first stage (Training BERT) and B from the second stage (Training Affective BERT), both with

the same dimension. While there are different ways to combine these two embeddings to achieve a meta-embedding [38, 94, 145], we combine these two contextual embeddings by a simple concatenation to obtain the final embedding $C = A \oplus B$ for a token.

6.2.2.3 Fine-tuning BERT Models

In this stage, the two trained BERT models and the trained AFE component are further combined by adding a fully-connected output layer with a softmax on top of the two BERT models. The [CLS] token representations from the two BERT models are fed into the output layer. This whole ACE 1 model is then fine-tuned with a labeled data set for sarcasm detection, in which all parameters are adjusted. After fine tuning, the two BERT models can be used to perform sarcasm detection given a new document.

6.2.3 Contextual Feature Embedding in ACE 2

Figure 6.2 illustrates the architecture of our second model (i.e., ACE 2). This model also contains 2 components: a) *affective feature embedding* (AFE), which is the same as the one in ACE 1 except that the input data are different, and b) *contextual feature embedding* (CFE), to be described in detail below. The purpose of ACE 2 is to avoid the time-consuming embedding-training with very large corpora. For this purpose, we use pre-trained BERT to obtain contextual embeddings, which is called the *feature-based* approach to using BERT [45]. For this purpose, we train the AFE component using the

texts in the downstream task (i.e., sarcasm detection) dataset, which is much smaller than the usual embedding-training corpus. Below we describe how the CFE component works and how the two components are combined.

6.2.3.1 Pre-trained Embeddings

In this stage, we use pre-trained BERT contextual embeddings (e.g., the output of the first BERT model in ACE 1, or any other pre-trained contextual embedding model) in the feature-based approach [45] to represent each input token/sentence generated from the hidden layers of the pre-trained model.

6.2.3.2 Obtaining Sentence Embeddings Using SBERT

The purpose of this stage is to obtain a sentence embedding given an input sentence. The most common approaches to derive a sentence embedding from a pre-trained BERT model is to i) average the outputs of the hidden layers or ii) using the output of the first special token [CLS] [112, 205, 208]. However, it has been shown that these methods produce poor sentence representations that are not semantically meaningful [153, 187]. This is because no independent sentence embeddings are computed in the BERT model, which makes it difficult to derive sentence embeddings from pre-trained BERT. Because of this, SBERT (a Sentence Transformer) [153] was proposed that uses a Siamese or triplet network structure to derive a sentence embedding using a pooling operation by i) computing the

mean of all output vectors (MEAN-strategy), or ii) computing a max-over-time of the output vectors (MAX-strategy) from the output of pre-trained BERT.

Given an input sentence, we first use the pre-trained BERT-large-uncased model to obtain the token embeddings, which are then passed to SBERT. SBERT computes a sentence embedding using the MEAN-strategy for the pooling operation to compute a sentence embedding, which is the default mode and was also suggested in [153] for classification tasks. For an input document, we concatenate the embeddings of all the sentences in the document to form a document representation.

6.2.3.3 Combining the Two Components

In this stage, a fully connected layer with a softmax is added on top of the CFE and AFE components. The input to the fully connected layer is the concatenation of the document embeddings from both CFE and AFE models (that is, the contextual embedding of a document from CFE and its affective feature embedding from AFE). The fully connected layer is trained as a classifier with the labeled dataset for sarcasm detection.

6.3 Experimental Evaluation

6.3.1 Corpora for Training Embeddings

The original BERT-Large-uncased was trained on 2.5 billion Wikipedia and 800 million BookCorpus words. Since BookCorpus is no longer publicly available, we used a news corpus along with Wikipedia to train BERT from scratch. For simplicity, through this chapter we called it **Wiki**[38], in which the news corpus consists of 142,546 articles from 15 American publications and Wikipedia consists of 23,046,187 articles from Wikipedia.

To test our models with text of less formal writing styles and more sarcasm occurrences, we also created another corpus and call it **WikiSarc** that contains Wiki and the following two datasets:

IMSDB: an Internet Movie Script Database[39], for which a scraper was used to retrieve comedy movie transcripts, resulting in 11.2 million movie transcripts, and **Riloff**: a dataset consisting of automatically extracted tweets: 60k containing the sarcasm hashtag and 100k random tweets using the method from [154].

[38]**News**:https://www.kaggle.com/snapcrack/all-the-news,**Wikipedia**:https://www.kaggle.com/jkkphys/english-wikipedia-articles-20170820-sqlite.

[39]**IMSDB**:https://www.imsdb.com/, **Scraper**:https://github.com/JoeKarlsson/movie-script-scraper

6.3.2 Affective Tasks Datasets

We evaluate our models on three datasets for sentiment analysis, three datasets for emotion detection in witch the details of the datasets are described in Section 4.3.3, and five labeled sarcasm detection datasets described in Table 6.1 as follows:

- **Onion**: This news headlines dataset[40] collected sarcastic versions of current events from The Onion[41] and non-sarcastic news headlines from HuffPost [119]. The dataset contains 28,619 headlines, with 13,634 labeled as sarcastic, and 14,985 as non-sarcastic.

- **IAC**: This is a subset of the Internet Argument Corpus [132]. The dataset contains response utterances annotated for the sarcasm detection task. We extract 3260 instances from the general sarcasm type, with 1630 as sarcastic and 1630 as non-sarcastic[42].

- **Reddit**: Self-Annotated Reddit Corpus (SARC)[43] is a collection of Reddit posts where sarcasm instances are labeled by authors (in contrast to other datasets where the data is typically labeled by independent annotators) [93]. This results in 1,010,826 posts, with 505,413 as sarcastic and 505,413 as non-sarcastic.

[40]https://github.com/rishabhmisra/News-Headlines-Dataset-For-Sarcasm-Detection

[41]https://www.theonion.com

[42]https://nlds.soe.ucsc.edu/sarcasm2

[43]https://nlp.cs.princeton.edu/SARC/2.0/pol/

Dataset	Genre	# Sarcastic	# Non-Sarcastic	Total	Max # of Sentences
Onion	News Headlines	13,634	14,985	28,619	5
Reddit	Reddit Forum	505,413	505,413	1,010,826	6
Pt′acek	Tweets	7,000	7,000	14,000	8
SemEval-2018	Tweets	2,396	2,396	4,791	7
IAC	Politic Debates	1630	1630	3,260	14

Table 6.1: Description of sarcasm detection datasets.

- **Pt′acek**: The dataset consists of manually annotated sarcastic tweets. Authors annotated 7k tweets as sarcastic and 7k ones as non-sarcastic [150].

- **SemEval-2018**: the dataset was provided in SemEval-2018 Task 3 (Irony detection in English tweets), with total of 4,792 tweets including 2,396 ironic and 2,396 non-ironic[180].

6.3.3 Experimental Setup

Both ACE 1 and ACE 2 models use a softmax at the output layer and cross-entropy is used as the loss function. The optimizer is Adam [96]. Parameter settings in the experiments are as follows:

Training: We train BERT-Large-uncased architecture for both stages of ACE 1 from scratch (24-layer, 1024-hidden, 16-heads, 340M parameters) , where the dimension is 768

and the max length for each sequence is 512. We experimented with different learning rates of $(1e-4, \beta1 = 0.9, \beta2 = 0.999)$, L2 weight decay of 0.01, and dropout probability of 0.1 for all layers as suggested for BERT-Large-Uncased [45].

Fine-Tune: It takes much less time to fine-tune our model than training it from scratch. In fact, the authors [45] recommend only 2-4 epochs of training for BERT fine-tuning on a specific NLP task with the learning rate Adam Optimizer: 5e-5, 3e-5, 2e-5, and batch size of 16, 32. Note that, all datasets are split into training/testing using 80%/20%. The results on test data are reported in F1-score, defined as $2\frac{p.r}{p+r}$, where p and r are precision and recall, respectively.

We train BERT from scratch on our datasets using Microsoft Azure ML[44] cluster of 8xND40-v2 nodes (64 NVidia V100 GPUs total) using the Microsoft CNTK parallelization algorithm[45], 16 TPUs (64 TPU chips), Tensorflow 1.15 with 1TB memory on Google Cloud and two 32-GPU clusters of V100/RTX 2080 Ti with 1TB memory. The training took up to 5 days.

[44]https://github.com/microsoft/AzureML-BERT

[45]https://docs.microsoft.com/en-us/cognitive-toolkit/multiple-gpus-and-machines

6.4 Discussion and Analysis

6.4.1 Comparing Variations of ACE 1 and ACE 2

In this section, we compare ACE 1 and ACE 2 to see which way of fusing the CFE and AFE components is more effective, investigate the performance of the models with and without considering affective features, and also investigate which training corpus (Wiki or WikiSarc) is more effective to train embeddings for sarcasm detection. Tables 6.2, §6.3 and §6.4 show the results of ACE 1 and ACE 2 on the 5 sarcasm datasets for different combinations of embedding-training corpus and affective feature representation methods. For example, in ACE 1 (Wiki) we train ACE 1 on Wiki followed by fine-tuning without incorporating affective features, while ACE 1 (Wiki)+(EAISe) means we train ACE 1 on Wiki and also incorporate the affective features of EAISe. For model ACE 2, the comparison is also among different pre-trained embeddings. For instance, ACE 2 (Wiki-BERT) + (EAISe) means the token embeddings inputted into SBERT in ACE 2 were from a BERT model pre-trained on our Wiki corpus and the affective feature representation in the AFE component is EAISe, while in ACE 2 (BERT) the embeddings inputted into SBERT are from the original pre-trained BERT (Large) model and the affective features are not used.

Corpus+Affective Feature	Onion	Reddit	Pt′acek	SemEval-2018	IAC
(Wiki)	83.88	80.25	71.06	77.38	85.10
(Wiki) + (EAISe)	<u>89.40</u>	<u>85.11</u>	**75.38**	**78.10**	<u>87.20</u>
(Wiki) + (EMoSi)	**90.07**	**86.20**	<u>75.18</u>	<u>77.91</u>	**88.40**
(WikiSarc)	90.61	86.59	75.15	80.45	89.20
(WikiSarc) + (EAISe)	<u>90.70</u>	<u>87.19</u>	<u>77.82</u>	<u>84.25</u>	<u>89.74</u>
(WikiSarc) + (EMoSi)	**92.21**	**89.22**	**80.71**	**84.57**	**93.14**

Table 6.2: F1-scores of model ACE 1 with different settings. Best results are in **bold** and 2nd best are <u>underlined</u>.

Results of 18 variations of the methods are shown in Table 6.2 and §6.3 .

As shown in Table 6.2 and §6.3, overall ACE 1 outperforms ACE 2, which suggests that incorporating affective features deeply in the embedding phase is more effective than combining the pre-trained contextual embeddings with affective feature embeddings later in the classification model. Also, including affective feature embeddings works better in both models than not including them. Furthermore, using WikiSarc (which contains more sarcastic texts) as embedding-training corpus leads to better results than using Wiki. The best performance of model ACE 1 is achieved by training it on WikiSarc with affective feature EMoSi. We call the resulting BERT model WikiSarcA-BERT (Sarcastic Affective

BERT). It is used as one of the pre-trained embedding models for ACE 2. Interestingly, when the models are trained on Wiki or the original BERT training corpus, there is no obvious winner between EAISe and EMoSi affective feature representation methods across the sarcasm datasets. But when WikiSarc is used as the training corpus, EMoSi is a clear winner for ACE 1 and EAISe is a clear winner on ACE 2. Our conjecture is that since WikiSarc has more sarcastic and emotional utterances than the general corpus such as Wikipedia, Book Corpus or News dataset, the affective feature representation method can make a difference in this case.

Results of 7 more variations are shown in Table 6.4. For example, in ACE 1 (Wiki)+(EAISe) we train ACE 1 on Wiki and also incorporate the affective feature of EAISe (only stage 2 of the model ACE 1) followed by fine-tuning. Note that, only stage 2 of the model ACE 1 means without concatenation the two pre-trained embeddings and CFE component of model ACE 1 starts from stage 2 in this experiment. Another example, for model ACE 2 (WikiSarcA-BERT) means the token embeddings inputted into SBERT in ACE 2 were from a BERT model pre-trained on our WikiSarc corpus in model ACE 1 (where the CFE component of the model ACE 1 starts from stage 2 and incorporate the affective feature of EMoSi) without incorporating affective features.

Corpus+Affective Feature	Onion	Reddit	Pt′acek	SemEval-2018	IAC
(BERT)	82.45	70.20	70.49	72.19	78.19
(BERT) + (EAISe)	84.11	**78.79**	**72.24**	76.19	**80.36**
(BERT) + (EMoSi)	**84.19**	77.45	71.85	**79.00**	80.00
(Wiki-BERT)	82.31	70.20	70.04	72.10	77.48
(Wiki-BERT) + (EAISe)	**84.33**	**79.22**	71.44	**79.61**	**80.25**
(Wiki-BERT) + (EMoSi)	84.00	77.04	**72.10**	79.15	79.88
(WikiSarc-BERT)	84.19	79.41	75.29	75.41	80.07
(WikiSarc-BERT) + (EAISe)	**87.46**	82.28	**79.09**	**80.46**	**85.29**
(WikiSarc-BERT) + (EMoSi)	86.17	**83.37**	78.19	80.11	85.10
(WikiSarcA-BERT)	87.09	82.25	76.84	78.18	84.79
(WikiSarcA-BERT) + (EAISe)	**90.31**	**86.50**	**80.39**	**84.33**	**88.19**
(WikiSarcA-BERT) + (EMoSi)	88.25	86.11	79.00	83.60	86.47

Table 6.3: F1-scores of model ACE 2 with different settings. Best results are in **bold** and 2nd best are underlined.

Models	Combos	Onion	Reddit	Pt′acek	SemEval-2018	IAC
ACE 1	(Wiki) + (EAISe)	89.41	85.61	**75.16**	**78.21**	87.00
	(Wiki) + (EMoSi)	**89.84**	**85.70**	74.34	77.92	**88.15**
	(WikiSarc) + (EAISe)	90.33	86.71	77.00	84.63	89.21
	(WikiSarc) + (EMoSi)	**92.00**	**89.01**	**80.54**	**84.31**	**93.37**
ACE 2	(WikiSarcA-BERT)	86.73	82.17	76.90	78.29	84.00
	(WikiSarcA-BERT) + (EAISe)	**90.00**	**85.88**	**81.11**	**84.15**	**87.27**
	(WikiSarcA-BERT) + (EMoSi)	88.02	85.47	79.30	83.19	86.73

Table 6.4: F1-score results of comparing different pre-trained embeddings with different affective embeddings for each model. The best score is highlighted in **bold**, and the second best result is underlined.

6.4.2 Evaluating Proposed Models against State-of-the-art Baselines

In this section we compare the performance of our proposed models against those of various state-of-the-art models in four different categories: (i) Only Affective, (ii) Only Contextual with Fine-Tune, (iv) Only Contextual with Pre-trained, and (iv) Affective Contextual. Some of the baseline results were taken from their original publication when the data split is the same as ours. In case we cannot find the result of a baseline for a data set that we use, the baseline was properly re-implemented using the available codes

Models	Onion	Reddit	Pt′acek	SemEval-2018	IAC
Rajadesingan et al., 2015 [151]	67.25	64.21	75.13	70.12	68.33
Ghosh et al., 2017 [59]	69.23	68.41	74.11	72.45	64.38
Hernandez farias et al., 2018 [76]	68.00	69.34	75.10	71.70	70.39
Zhang et al., 2019 (a) [204]	-	-	69.15	64.28	-
Zhang et al., 2019 (b) [204]	-	-	72.39	65.33	-
Zhang et al., 2019 (c) [204]	-	-	72.47	67.55	-
AFE with EAISe	70.49	71.87	**76.90**	72.51	72.40
AFE with EMoSi	**74.20**	**74.04**	76.40	**72.60**	**73.01**

Table 6.5: F1-scores for comparing our models against state-of-the-art models (Only Affective). The best scores are in **bold**, and 2nd best are underlined, while the 3rd best are double underlined.

and guidelines. The model in [204] in the "Only Affective" category was only designed for tweets with hashtags. Thus, its results are reported only on two datasets that were from Twitter. Also, the model in [74] in the "Affective Contextual" category was not possible to be re-implemented for two of our datasets because the user history information embedding used in their models was not available in these datasets.

(i) **Only Affective**: We compare the AFE component of our models with those that only used hand-crafted or automatic features for sarcasm detection. To make a fair

Models	Onion	Reddit	Pt′acek	SemEval-2018	IAC
Potamias et al., 2019 [149]	84.39	78.00	71.01	70.00	85.21
RoBERTa	80.23	76.04	67.25	68.00	82.44
XLNet-Large	79.66	76.48	69.33	68.25	70.06
BERT-Base	80.04	76.14	67.13	69.03	82.27
BERT-Large	83.49	78.21	70.33	76.19	84.25
ACE 1 (WikiSarc)	**90.61**	**86.59**	**75.15**	**80.45**	**89.20**

Table 6.6: F1-scores for comparing our models against state-of-the-art models (Only Contextual with Fine-Tune). The best scores are in **bold**, and 2nd best are underlined, while the 3rd best are double underlined.

comparison, our AFE component is trained on the sarcasm dataset (not the Wiki or WikiSarc corpus which would produce better results), to be the same as the baseline methods. The details of each baseline is as follows:

- **Rajadesingan et al., 2015** [151]: Authors proposed a behavioral modeling framework for sarcasm detection. They discussed different forms of sarcasm: *i*) a contrast of sentiments, *ii*) a complex form of expression, *iii*) a means of conveying emotion, *iv*) a possible function of familiarity, and *v*) a form of written expression. They constructed relevant features to detect these forms on the Twitter dataset.

Models	Onion	Reddit	Pt′acek	SemEval-2018	IAC
Zhang et al., 2016 [202]	67.08	69.20	70.49	70.66	69.38
Ilič et al., 2018 [84]	70.12	76.05	75.46	68.90	72.00
RoBERTa	76.51	66.00	62.51	66.37	75.10
XLNet-Large	79.23	70.25	60.13	66.45	72.41
BERT-Base	78.13	66.27	63.12	68.14	74.90
BERT-Large	79.11	65.27	62.39	69.47	75.48
ACE 2 (WikiSarcA-BERT)	**87.09**	**82.25**	**76.84**	**78.18**	**84.79**

Table 6.7: F1-scores for comparing our models against state-of-the-art models (Only Contextual with Pre-trained without Fine-Tune). The best scores are in **bold**, and 2nd best are underlined, while the 3rd best are double underlined.

- **Ghosh et al., 2017** [59]: They investigated several Long Short-Term Memory (LSTM) networks variations that can model both the conversation context and the sarcastic response. They showed that the conditional LSTM and LSTM networks, with sentence-level attention on context and response, outperform the LSTM model that only reads the response.

- **Hernandez farias et al., 2018** [76]: They proposed a model exploring the utility of *affective features* based on a wide range of lexical resources (available for English)

in the sarcasm detection task.

- **Zhang et al., 2019 (a)** [204]: In their 1'st model, authors proposed a Sentiment-Augmented Attention Bi-LSTM model employing an attention-based neural network to identify context incongruity in the irony detection task. They used sentiment word corpora as external features, and designed a soft attention mechanism focusing on context incongruity of tweets.

- **Zhang et al., 2019 (b)** [204]: In their 2'nd model, authors improved the attention mechanism in a supervised manner to capture the context of incongruity in Twitter data. In particular, they incorporated two different types of sentiment resources (sentiment word lexicon and sentiment tweets copra) into the irony detection task.

- **Zhang et al., 2019 (c)** [204]: In their 3'rd model, authors proposed a model to transfer deep features from sentiment analysis into the irony detection task for learning both explicit and implicit context incongruity in Twitter data. Their model consists of two Bi-LSTMs: one Bi-LSTM model serves as the sentiment feature extractor while the other one acts as the irony detector.

Table 6.5 shows AFE with EMoSi performs the best on 4 of 5 datasets, while the second best is AFE with EAISe. Note that the AFE results in this experiment are not as good as the ACE 2 results in Table 6.3, indicating combining contextual and affective features is better than using the affective features alone.

(ii) **Only Contextual with Fine-Tune**: We compare the CFE component of ACE 1 without using affective features (i.e., the stage 1 model) with those that initialize the model by pre-trained embeddings followed by fine-tuning. Among the baselines, [149] is a transformer-based model for sarcasm detection. The other baselines are benchmark models used for a wide range of NLP tasks. The details of each baseline is as follows:

- **Potamiad et al., 2019** [149]: They proposed a model (i.e., RCNN-RoBERTa) leveraging the pre-trained RoBERTa model and a recurrent convolutional neural network to tackle figurative language in sarcasm/irony detection in social media [149].

- **RoBERTa**: A Robustly Optimized BERT Pre-training approach proposed by [107]. The model uses dynamic masking instead of static masking (that was used in BERT), and are optimized to improve the accuracy of different NLP tasks (e.g., question answering) on GLUE, RACE and SQuAD detests.

- **XLNet-Large**: A generalized auto-regressive pre-training method that uses a permutation language modeling objective to combine the advantages of AR (Auto-Regressive) and AE (Auto-Encoding) methods. The model fixes BERT's negligence on dependency between the masked positions, causing a pre-train/fine-tune discrepancy [198].

- **BERT-Base/Large**: A Bidirectional Encoder Representations from Transformers.

The model is trained on an unlabeled text corpus over two unsupervised tasks: *i*) *Masked Language Model* (MLM), in which, some of the tokens in the input sequences are masked and the model is trained to predict these masked tokens, and *ii*) *Next sentence Prediction* (NSP), where the model receives pairs of sentences as input and learns to predict if the second sentence in a pair is the subsequent sentence of the first one in the training corpus [45].

Table 6.6 shows that ACE 1 trained on WikiSarc significantly outperforms all baselines on all five datasets, indicating that training embeddings with a corpus that contains more emotional or sarsastic untarrences is better for sarcasm detection.

(iii) **Only Contextual with Pre-trained without Fine-Tune**: We compare the CFE component of ACE 2 without incorporating affective features with those that used either contextual pre-traind embeddings (i.e., transformer-based models) or other pre-trained embeddings (e.g. GloVe used in [202]) without fine-tuning the embeddings. The details of baseline models are as follows:

- **Zhang et al., 2016** [202] : Authors proposed a deep neural network using a gated recurrent neural network (GRNN) to induce semantic features for sarcasm detection. In particular, they modeled the tweet content with a GRNN, and used a gated pooling function to extract features, then predicted sarcastic tweets.

Models	Onion	Reddit	Pt′acek	SemEval-2018	IAC
Poria et al., 2016 [148]	70.00	64.27	67.00	69.45	60.25
Amir et al., 2016 [10]	67.79	65.14	69.25	71.59	68.51
Yang et al., 2016 [199]	63.25	64.83	71.16	67.45	70.14
DeepMoji, 2017 [53]	69.47	53.08	63.51	69.27	71.00
Wu et al., 2018 [189]	70.00	69.20	68.50	71.20	65.23
Tay el al., 2018 (a) [174]	70.68	67.25	71.52	70.01	72.00
Tay el al., 2018 (b) [174]	70.13	68.23	70.13	69.46	71.85
Hazarika et al., 2018 [74]	70.90	75.16	70.24	-	-
Kumar et al., 2020 [99]	68.36	77.01	70.27	75.44	69.33
ACE 1 (WikiSarc) + (EMoSi)	**92.21**	**89.22**	**80.71**	**84.57**	**93.14**
ACE 2 (WikiSarcA-BERT) + (EAISe)	90.31	86.50	80.39	84.33	88.19

Table 6.8: F1-scores for comparing our models against state-of-the-art models (Affective-Contextual). The best scores are in **bold**, and 2nd best are underlined, while the 3rd best are double underlined.

- **Ilič et al., 2018** [84]: They proposed a deep learning model based on character-level word representations obtained from ELMo [143]. The model used a learned representation features derived from morpho-syntactic cues.

Table 6.7 shows that ACE 2 with the pre-trained embedding model WikiSarcA-

BERT consistently outperforms all the baselines. This finding supports our intuition that incorporating the affective feature information into the contextual word embeddings in the training phase (ACE 1) improves the performance in sarcasm detection, as WikiSarcA-BERT was trained in stage 2 of ACE 1.

(iv) **Affective-Contextual**: We compare our ACE 1 and ACE 2 models that combine the AFE and CFE components with those that used both affective features and pre-trained embeddings in a single architecture for sarcasm detection. The details of baselines are as follows:

- **Poria et al. 2016** [148]: They developed pre-trained sentiment, emotion and personality models for identifying sarcastic text using Convolutional Neural Networks (CNN-SVM).

- **Amir et al., 2016** [10]: Authors proposed a deep neural network model (called CUE-CNN) that learns embeddings of content with lexical signals to recognize sarcasm in text documents.

- **Yang et al., 2016** [199]: They proposed an attention-based neural model that learns an intra-attentive representation of the sentence, enabling it to identify contrasting sentiment, situations and incongruity for sarcasm detection.

- **DeepMoji**: These word embeddings were trained using BiLSTM on 1.2 billion tweets with emojis [53]. The model learns representations of emotional content in

texts.

- **Wu et al., 2018** [189]: Authors proposed a system based on a densely connected LSTM network (every LSTM layer will take all outputs of previous layers as inputs) with a multi-task learning strategy to combine the information in different tasks. The model improves the performance using POS tags and sentiment features.

- **Hazarika et al., 2018** [74]: They proposed a ContextuAl SarCasm DEtector (CASCADE) by adapting a hybrid approach of both content-based and context-driven modeling for sarcasm detection. They used user profiling along with discourse modeling from comments in discussion threads. Then, the information is used jointly to learn a CNN-based model.

- **Tay el al., 2018 (a)** [174]: Authors proposed a model called "MIARN" that utilizes a multi-dimension intra-attention mechanism to overcome limitations of sequential neural networks in capturing words' incongruities in sarcasm detection.

- **Tay el al., 2018 (b)** [174]: In another model, they proposed a model called "SIARN" which employs a single-dimension intra-attention network for irony detection.

- **Kumar et al., 2020** [99]: Authors proposed a model that uses the semantic, sentiment and punctuation based hand-crafted features for sarcasm detection. They utilized multi-head attention based Bidirectional Long-Short Term Memory (MHA-BiLSTM) combined with Glove Pre-trained embeddings for this purpose.

The results in Table 6.8 showed that ACE 1 with WikiSarc and the EMoSi affective feature representation outperforms all the baselines, while the second best is our ACE 2 (WikiSarcA-BERT)+(EAISe) on all 5 datasets. Note that none of the existing baselines in this category uses contextual embeddings. Thus, the results suggest that using transformer-based models to generate contextual embeddings leads to better performance.

6.5 Evaluating the Performance of Proposed Models on Other Affective Tasks

In this section, we plan to investigate the performance of the proposed affective and contextual models (ACE 1 and ACE 2), on other tasks, such as emotion detection and sentiment analysis. The details of the datasets are described in Section 4.3.3. Furthermore, we form a comparative study of the accuracy performance of previous model using pre-processing for affective tasks with the current proposed models.

Table 6.9 and Table 6.10 show the results of ACE 1 and ACE 2 on three datasets for sentiment analysis and another three datasets for emotion detection with different combinations of embedding-training corpus and affective feature representation methods. For example, in ACE 1 (Wiki) we train ACE 1 on Wiki followed by fine-tuning without incorporating affective features, while ACE 1 (Wiki) + (EAISe) means we train ACE 1 on Wiki and also incorporate the affective features of EAISe. For model ACE 2,

the comparison is also among different pre-trained embeddings. For instance, ACE 2 (WikiSarc-BERT) + (EAISe) means the token embeddings inputted into SBERT in ACE 2 were from a BERT model pre-trained on our WikiSarc corpus and the affective feature representation in the AFE component is EAISe.

As shown in Table 6.9 and Table 6.10, overall ACE 1 outperforms ACE 2 for the task of sentiment analysis, which suggests that incorporating affective features deeply in the embedding phase is more effective than combining the pre-trained contextual embeddings with affective feature embeddings later in the classification model. These findings are consistent with the findings in Tables §6.2 and §6.3 on sarcasm detection datasets. However, in the case of emotion detection we observe that overall ACE 2 outperforms ACE 1, which suggests that in multi-class classification, incorporating affective features in training phase and fine-tuning have less effects in the performance. We also need to mention that our emotion detection datasets are relatively smaller than the other two tasks datasets.

Moreover, as it's observed in both Tables 6.9 and 6.10, including affective feature embeddings works better in both models than not including them. Furthermore, using WikiSarc as embedding-training corpus in most cases leads to better results than using Wiki. The best performance of model ACE 1 is achieved by training it on WikiSarc with affective feature EMoSi for both datasets. We call the resulting BERT model WikiSarcA-BERT (Sarcastic Affective BERT). It is used as one of the pre-trained embedding models

for ACE 2.

Interestingly, as it was observed in sarcasm detasets when the models are trained on Wiki, there is no obvious winner between EAISe and EMoSi affective feature representation methods across the sentiment analysis datasets, while for emotion detection datasets best results achieved training wiki with affective feature of EAISe in two datasets out of three. In addition, as the training corpus, EMoSi is a clear winner for both models on both datasets.

Corpus+Affective Feature	**Sentiment Analysis**			**Emotion Detection**		
	IMDB	**Semeval**	**Airline**	**Alm**	**ISEAR**	**SSEC**
(Wiki)	93.00	75.60	90.33	63.40	70.00	69.88
(Wiki) + (EAISe)	<u>93.76</u>	**78.70**	<u>91.69</u>	**65.06**	**71.80**	<u>70.08</u>
(Wiki) + (EMoSi)	**95.28**	<u>77.30</u>	**93.80**	<u>64.61</u>	<u>70.73</u>	**71.14**
(WikiSarc)	95.00	78.80	94.87	65.70	70.29	70.16
(WikiSarc) + (EAISe)	<u>96.19</u>	<u>78.17</u>	<u>94.10</u>	**67.29**	**73.80**	**74.05**
(WikiSarc) + (EMoSi)	**97.13**	**80.67**	**96.25**	<u>66.29</u>	<u>72.20</u>	<u>73.15</u>

Table 6.9: F1-scores of model ACE 1 with different settings on other affective tasks. Best results are in **bold** and 2nd best are <u>underlined</u>.

Corpus+Affective Feature	Sentiment Analysis			Emotion Detection		
	IMDB	**Semeval**	**Airline**	**Alm**	**ISEAR**	**SSEC**
(WikiSarc-BERT)	85.61	73.68	88.70	64.30	70.15	69.27
(WikiSarc-BERT) + (EAISe)	87.20	74.60	**91.22**	65.49	**72.80**	70.63
(WikiSarc-BERT) + (EMoSi)	**88.10**	**75.00**	90.37	**66.07**	71.68	**72.29**
(WikiSarcA-BERT)	89.37	75.86	91.50	67.40	73.40	74.00
(WikiSarcA-BERT) + (EAISe)	91.60	76.00	93.15	68.12	74.06	**76.60**
(WikiSarcA-BERT) + (EMoSi)	**94.30**	**78.25**	**95.02**	**70.31**	**75.60**	75.39

Table 6.10: F1-scores of model ACE 2 with different settings on other affective tasks. Best results are in **bold** and 2nd best are underlined.

Table 6.11 demonstrates the results of comparing the proposed models ACE 1 and ACE 2 against the customized pre-processing model that was proposed in the previous chapter 5. As can be seen, overall ACE 1 outperforms the ACE 2 and pre-processing model on all 6 datasets for sentiment analysis and sarcasm detection. While, for emotion detection dataset ACE 2 performers better than ACE 1 and pre-processing method.

Models	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
Pre-processing	94.22	75.20	94.88	80.21	80.34	67.41	63.10	72.66	72.80
ACE 2	94.30	78.25	95.02	88.19	90.31	86.50	**70.31**	**75.60**	**75.39**
ACE 1	**97.13**	**80.67**	**96.25**	**93.14**	**92.21**	**89.22**	67.29	73.80	74.05

Table 6.11: F-score Results of comparing ACE 1 and ACE 2 against the customized pre-processing model.

6.6 Summary

We proposed two novel models (ACE 1 and ACE 2) that incorporate contextual and affective features in a deep neural network architecture for sarcasm detection. Each model extends BERT's architecture by incorporating into it affective features. ACE 1 uses them to adjust the contextual embeddings and also fine-tune the model, while ACE 2 uses them along with SBERT for performing the final classification. Our evaluation results showed that combining the two types of features greatly improves the sarcasm detection accuracy. In particular, deeply incorporating the affective features in the embedding training process (as in ACE 1) is more beneficial than simply concatenating the two types of features (as in ACE 2). We also observed that training embeddings with corpora containing rich sarcastic or emotional utterances greatly benefits the sarcasm detection tasks. Our findings suggest that transformer-based models like BERT can be trained to

incorporate task-specific features to improve downstream task performance. Our results demonstrated that the proposed models can improve the performance of other tasks, such as emotion detection and sentiment analysis. However, in the case of emotion detection, overall ACE 2 outperforms ACE 1, which suggested that the proposed model ACE 1 with fine-tuning with BERT are more successful in binary classification such as sentiment analysis and sarcasm detection than in multi-class classification.

7 Affective and Contextual Embedding Model for Feature Representation Learning in Affect-Aware Recommendation

7.1 Introduction

Affects greatly influence the human behaviours and choices [68, 162] according to cognitive psychology and are widely recognized as main factors in decision-making [103], particularly when spontaneous decisions are taken as in news portals or music streams where users typically interact with items at a fast pace. However, affects are also very difficult to detect, quantify and measure precisely, which is one of the primary reasons for a relatively small number of previous work on using affects in recommender systems. In this chapter, we are filling the gap by presenting the Affect-Aware Recommendation (AARec) model, a recommendation system capable of incorporating affects into personalized recommendations of news and music items. In particular, the AARec recommender engine can incorporate the affective information into its algorithm in order to serve more

relevant and engaging recommendations for further reading/listening. In other words, incorporating affective information of news articles and music lyrics into the system allows the recommendation engine to better estimate the similarity between users/items, items/items, and to provide more targeted recommendations as a result. However, incorporation of implicit affective information from items into the recommendation workflow is not a trivial task and at least two questions arise, which we investigate in this chapter as follows: (i) *which affect detection approaches are more beneficial to extract affective information for recommender systems (RS)?*, and (ii) *how do we incorporate the affective information into the recommendation algorithm?*

Leading up to this chapter, we have introduced and analyzed numerous models for enhancing affect analysis in text, particularly through the lens of sentiment analysis, emotion detection and sarcasm detection. Although usable as stand-alone affect detection systems, various application systems can be further enhanced by these affective models, such as recommender systems. In this chapter, we first describe the general recommendation algorithms, then we discuss the affective and contextual feature representation in recommendation models. Further, we present an affect-aware recommendation model describing the application of our affect detection models in a non-affective framework of recommendation system to demonstrate the usefulness of the proposed models in tasks beyond affect detection. Specifically, we leverage affective and contextual embedding models (described earlier in chapter 6) as well as pre-processing factors (described earlier

in chapters 4 and 5) in an affect-aware recommendation to demonstrate the usefulness of our proposed affective models in recommender systems. Further, we present a comparative study measuring the performance of EmoRec (proposed model from chapter 3) in which we used various affect detection models for affective feature extraction against Affect-Aware Recommendation (AARec). Interestingly our results demonstrate that improved affect detection models proposed in previous chapters could improve the performance of recommendation models.

The chapter is structured as follows. In section 7.2, a general discussion on recommendation algorithms is presented. Then, affective and contextual feature representation is presented in section 7.3 for recommendation models as well as the framework of the affect-aware recommendation model. Experimental evaluation is described in section 7.4, and we demonstrate the results in section 7.5. Section 7.6 concludes the chapter with key insights of the research.

7.2 Recommendation Algorithms

Predicting future behaviours of users using their historical information is the foundation of many recommendation systems. Past behaviour sequences usually represent the individual's next action, which is both intuitive and reliable [203]. Also, substantial history of users' activity (e.g. click and search history) represent users' rich consumption patterns. As illustrated in Figure 7.1, user interaction behaviours inherently form a sequence over

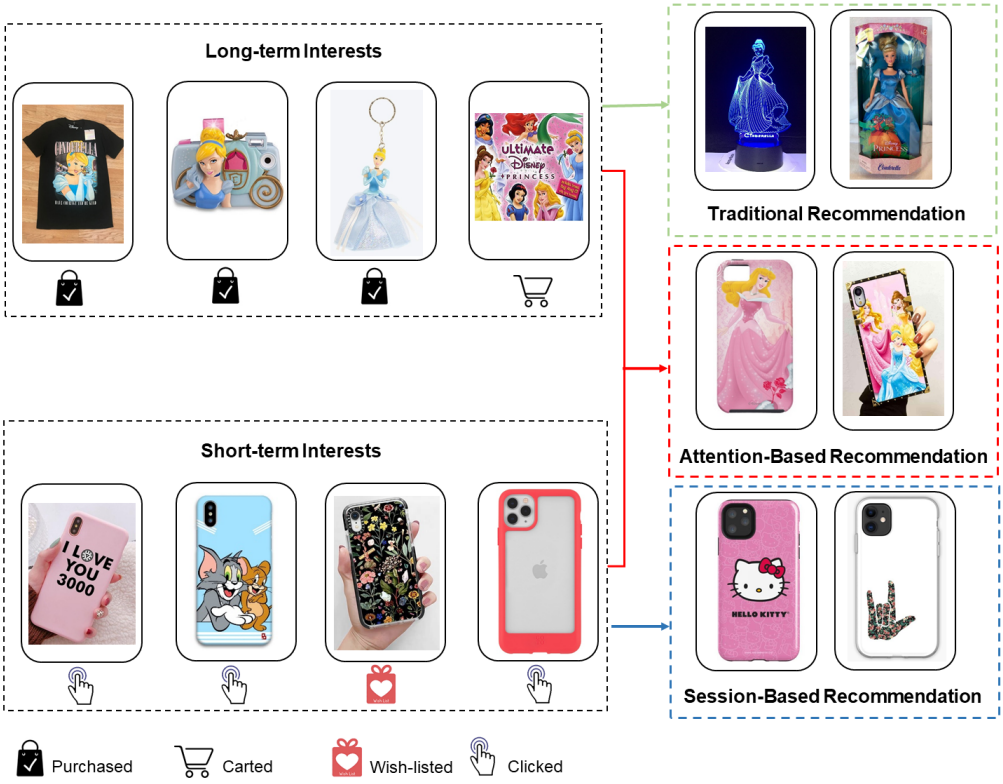

Figure 7.1: Different Recommendations Based on The Recommendation Model Algorithm.

time that increasingly exposes the long-term historical desires of users and short-term customers' motives, which is a typical online shopping behaviour of a user. As it is observed in Figure 7.1, traditional recommendations (*green box*) such as Matrix Factorization (MF) [171] or collaborative filtering models mostly consider the overall long history of users when recommending items. While, if only current user behaviour (short-term interests) sequences are considered, session-based models such as Recurrent Neural Networks (RNN) [78] are used to suggest items (*blue box*). However, with attention-based recommendations (*red box*), the attention can be focused on both interaction behaviour

by considering user's long-term preferences as well as current consumption preferences. For instance, the user behaviour illustrated in Figure 7.1 demonstrates that the user is a fan of "Cinderella" and "Disney Princess". Considering the long-term interests of the user using a traditional recommendation will result in recommending a "Cinderella Doll" or a "Cinderella Crystal Light", while consideration of the short-term interests of colorful or cartoon phone-cases will result in suggesting the "Hello Kitty" phone-case using session-based recommendations. However, if we would like to consider both users' long and short term interests using attention-based recommendation, we can suggest a phone-case with "Disney Princess" characters.

Users' interests in real life change over time and how to appropriately combine historical preferences with current consumption preferences is not a trivial task. In addition, most existing recommendation models ignore the mining and representation of affective features while suggesting items to the users and instead they use various affect detection approaches for affect feature extraction, which is a time-consuming task. Thus, we are presenting an Affect-Aware Recommendation (AARec) model with an attention-based algorithm that incorporates affective and contextual features of items using the affective contextual embedding models as well as pre-processing models for affect detection presented in the previous chapters for news and music recommendations.

7.3 Affective and Contextual Feature Representation in Recommendation Models

In recent years, deep neural network–based algorithms have been extensively studied, and many deep learning models have been proposed. The main advantage of these models over traditional models is that they do not require lots of manual feature engineering, which is the most challenging part of designing machine learning systems. In addition, they improve the ability of the model to leverage the context information. Initially, word embedding models [116] are proposed as a method to represent the word in a continuous way to better support neural network structure. Distributed word representations are an effective and compact way to represent text and are commonly used for various NLP tasks. The research community has also studied them in the context of many other machine learning models, such as recommender systems, where they are typically used as features [97]. Then several new neural network structures, including recurrent neural networks (RNNs) [78] such as long short-term memory (LSTM) [66], have been introduced to better represent sequence-based inputs and overcome long-term dependency issues. Recently, contextual word representations generated from pre-trained bidirectional language models (e.g. BERT, RoBERTa) have been shown to significantly improve the performance of state-of-the-art systems [45, 107].

For learning vector-space representations of items for a recommendation system, we

first order all the clickstream events (e.g. all the items that are accessed by a user) per user sorted by its timestamp into a sequence of items. Then, we map the content of each item (e.g. news articles or lyrics of the songs) into their vector representation using our proposed models affective and contextual embedding ACE 1 and ACE 2 from chapter 6 as following:

- Each input document is first chunked into sentences.
- Given an input sentence, we first use the pre-trained BERT model that was trained with ACE 1 to obtain the token embeddings, which are then passed to SBERT from Model ACE 2. SBERT computes a sentence embedding using the MEAN-strategy for the pooling operation to compute a sentence embedding.
- We concatenate the embeddings of all the sentences in the document to form a document representation of each item that was accessed by a user.

7.3.1 Affect-Aware Recommendation Model (AARec)

In order to utilize affective information from our proposed models ACE 1 and ACE 2 in recommendation algorithms, we first embed sparse user and item interaction information into vector representation (which was explained in the previous section), which assign each user or item an affective contextual representation using ACE 1 and ACE 2 model instead

of the basic index or one-hot encoding [46]. As such, by having the vector representation of each item that was accessed by a user in a sequence, we can efficiently utilize different recommendation algorithms to make the prediction. Therefore, let U denote a set of users and I denote a set of items, in which $|U|$ and $|I|$ are representing the total number of users, and items respectively. For each user $u \in U$, his/her sequential transactions (clickstream data) or sessions are denoted as $L^u = \left\{S_1^u, S_2^u, ..., S_T^u\right\}$, where T is the total number of time steps and $S_t^u \subseteq I(t \in [1, T])$ demonstrates the item set corresponding to the transaction of user u at time-step t. In addition, for a fixed time-step t, the item set S_t^u indicates user u's short-term interest at time t, while the set of items which were accessed before the time-step t demonstrates the user's long-term interest, which is denoted as $L_{t-1}^u = S_1^u \cup S_2^u \cup ... \cup S_{t-1}^u$. Hence, given users and their sequential transactions in vector representation as L, we aim to predict the next items that the users will access based on learned preferences from L.

[46] One hot encoding is a technique that converts categorical variables to numerical in an interpretable format to quantify categorical data. In particular, this method produces a vector with a length equal to the number of categories in the dataset.

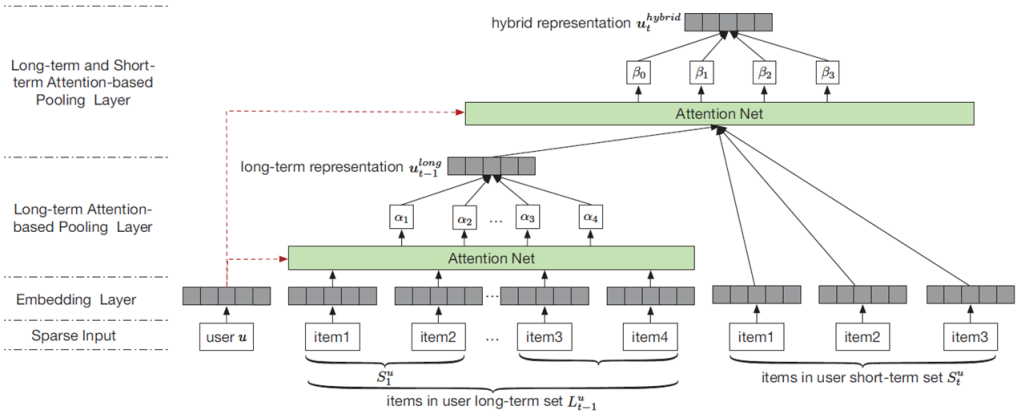

Figure 7.2: The Architecture of the Sequential Hybrid Attention-Based Model (SHAN) Proposed in [200]

Inspired by the model proposed in [200], we design an affect-aware recommendation (AARec) model by adopting this attention-based model. This model is a sequential hierarchical attention network (SHAN) for next item prediction that employs two attention networks to mine long and short term preferences of users [200], as is demonstrated in Figure 7.2. The model considers not only dynamic properties in user's long and short term preferences, but also high-level complex interactions between user and item factors as well as item and item factors. In contrast with SHAN, which obtains the vector representation of user/items by one-hot encoding, we represent each item accessed by a user using our affective and contextual feature representation that was explained in section 7.2. Then the hybrid representation for each user is generated through jointly

learning long and short term preferences. Further, to capture the long-term preference before time step t, the model learns a long-term user representation, that is a weighted sum over the embeddings of items in the long-term item set L^u_{t-1}, while the weights are calculated by an attention-based pooling layer driven from the user embedding [200]. Further, the final hybrid user representation combines the long-term user representation with the embedding of the items in the short-term item set to better integrate the short-term preference, where the weights are learned by another attention-based pooling layer. The model simultaneously incorporates the dynamic long and short term user preferences, since they can have different effects on the next item to be accessed by the user when they are using different learned weights by attention mechanism [183]. Moreover, since the user 's long-term item set typically changes over time, obtaining a static representation of long-term preference for each user may not completely express the dynamic of long-term user preference [200]. Therefore, it can be more beneficial to reconstruct the long term user representation from the up-to-date long term item set. The attention mechanism [183] computes the importance of each item in the long-term item set for a given user, then it aggregates the embeddings of these items to form the long term user preference representation. Further, by assigning weights to long-term representations and embeddings of items in the short-term item set, we can capture the high-level representation of users in the long and short term attention-based pooling layer.

7.4 Experimental Evaluation

We conduct our experiments considering the findings of overall experiments up to this chapter. In particular, the best proposed models from the previous chapters will build up the variations of our current experiments. To have an affective and contextual vector representation of an item's content (e.g. News articles and lyrics from the songs) using ACE 1 and ACE 2, which are accessed by a user, we consider the following modes:

Mode 1: We utilize the embeddings generated from model ACE 1 WikiSarc-BERT without incorporating any affective features during training. Then, we obtain a document embedding of each item using the ACE 2 model by using the pre-trained WikiSarc-BERT model to obtain the token embeddings, which are then passed to SBERT. SBERT computes a sentence embedding, and by concatenating the embeddings of all the sentences in the document, we will get the document representation of each item.

Mode 2: Similar to Mode 1, a document representation will be obtained from ACE 2 as it was described in the previous Mode, except this time we use pre-trained embeddings WikiSarcA-BERT by training ACE 1 and incorporating affective feature of EMoSi deeply in the embedding phase.

Mode 3: We obtain a document representation from model ACE 2 using pre-trained embeddings WikiSarcA-BERT (incorporating affective feature of EMoSi) from ACE 1 similar to Mode 2, but then we concatenate the document embeddings with its affective

feature embedding from AFE incorporating (e.g. the first component of model ACE 2) using the affective feature of EMoSi.

Mode 4: In this mode, we consider the customized pre-processing factors (`c-pre`), which was proposed in training word embeddings for BERT model in Chapter 5 to train the model ACE 1 (WikiSarcA-BERT) incorporating affective feature of EMoSi. Further, a document representation will be obtained from ACE 2 with the concatenation of the affective feature EMoSi.

7.4.1 Datasets

Our experiments are conducted on real-world datasets from two domains: news and music which are explained in detail in Section 3.4.1.

7.4.2 State-of-the-art Recommendation Algorithms

We devise state-of-the-art recommendation models to investigate *whether*, *how* and *to what extent* affective and contextual feature embeddings generated from ACE 1 and ACE 2 can improve recommendations and compare their results with AARec. A brief description of these three models is as follows:

- **GRU4Rec**: This is a session-based recommendation proposed in [78] based on Gated Recurrent Unit (GRU) with Recurrent Neural Networks (RNN). The model uses RNN to capture sequential dependencies to make predictions.

- **Caser**: Convolutional Sequence Embedding Recommendation (Caser) model that is a sequential recommendation method based on a Convolutional Neural Network (CNN) to learn users' sequential patterns for sequential recommendation [173]. This model incorporates the convolutional neural network with a latent factor model focusing on long term user preferences.

- **RCNN**: A recurrent convolutional neural network model proposed in [192], which not only utilizes the recurrent architecture of RNN to capture the long-term dependencies, but also leverages the convolutional function of Convolutional Neural Network (CNN) model to extract the short-term sequential patterns among recurrent hidden states.

7.5 Discussion and Analysis

7.5.1 Evaluating the Effects of Proposed Affect Detection Methods in Recommendation Algorithms

The performance of the various modes in recommendation algorithms, summarized in Table 7.1 and Table 7.2, is reported in terms of F-score on two datasets. In this section, we investigate whether the application of proposed affect detection models on 4 different recommendation algorithms may result in better performance by considering different combinations of embedding-training corpus and affective and contextual feature represen-

tation methods along with pre-processing factors on News and Music recommendations. We also evaluate the performance of the models with and without considering affective features. For example, in Mode 1, we do not consider any affective features either during training ACE 1 to generate the embeddings or getting the document representation using ACE 2. Results of 4 variations of the methods in 4 modes are shown in Table 7.1 and Table 7.2 for two recommendation datasets as Music and News.

As shown in Table 7.1 and Table 7.2, overall Attention-Based model (AARec) outperforms the other three models over 4 modes for both Music and News recommendations, which suggests that considering both long and short term interests of users utilizing attention mechanism yields better results than only considering either of them. In addition, our findings are consistent with what was also mentioned in [200] that since user preference is dynamic at different time steps, different items have different effects on the next item that will be accessed by users. Also, for different users, same items may have different influences on the next item prediction, and hence a hierarchical attention network could be beneficial to improve the performance. We called the resulting attention-based model in Mode 4 as Affective-Aware Recommendation (AARec) Model.

Models	Mode 1	Mode 2	Mode 3	Mode 4
GRU4Rec	70.23	72.08	76.04	80.30
Caser	70.18	71.81	75.33	79.25
RCNN	73.49	74.20	73.68	81.29
AARec	**78.49**	**79.74**	**80.20**	**83.62**

Table 7.1: Comparison of The performance of Different Recommendation Models Using Different Modes on Music Dataset (F-score)

Models	Mode 1	Mode 2	Mode 3	Mode 4
GRU4Rec	73.68	74.02	76.60	80.46
Caser	72.29	73.07	75.00	76.59
RCNN	76.15	78.37	79.60	83.70
AARec	**80.19**	**81.47**	**84.20**	**86.09**

Table 7.2: Comparison of The performance of Different Recommendation Models Using Different Modes on News Dataset (F-score)

Interestingly, as it's demonstrated in Tables 7.1 and 7.2, considering affective features either by incorporating affective features deeply in the embedding phase (Mode 2) or combining the affective pre-trained contextual embeddings (WikiSarcA-BERT) with affective

feature embeddings later (Mode 3 and Mode 4) is more beneficial than not considering any affective features (Mode 1). Moreover, our analysis also reveals that applying customized pre-processing for training ACE 1 (Mode 4) improves all recommendation methods' predictive performance.

7.5.2 Evaluating the Performance of AARec Against EMoRec

In this section, we compare the performance of using various affective feature extraction methods in a recommendation model EMoRec that was proposed in Chapter 3 against the proposed affective contextual feature extraction in a recommendation model AARec on both Music and News datasets. Also, we investigate whether the use of affective features in both models will boost the performance of recommendation.

As illustrated in Table 7.3, AAREC significantly outperforms EMOREC on both datasets, which suggests that not only using proposed affect detection models yields better results than using various affective feature extraction methods, but also considering affective features in both models is more beneficial than not including them, emphasizing the importance of improved affect detection models in general for recommendation tasks.

.

Dataset	Model	Non-Affect	Affective
Music	EMOREC	73.68	76.06
	AAREC	**78.49**	**83.62**
News	EMOREC	78.20	80.30
	AAREC	**80.19**	**86.09**

Table 7.3: F-score Comparison of EMOREC Performance Against AAREC on News and Music Dataset

7.6 Summary

This chapter demonstrates the usefulness of affective information in the form of affective and contextual representation utilizing our proposed affect detection models on four different recommendation algorithms. The proposed application of affective and contextual embedding along with pre-processing factors outperforms several baselines demonstrating that the proposed affect detection approaches can be effectively applied to solve problems beyond affect detection such as in recommendations. Our analysis revealed that including affective features in recommendations improved the predictive performance for all the recommendation methods in general, while the best performance was achieved by using attention-based model in Mode 4 (AARec). Moreover, comparing the results of the

AARec model against EMoRec indicated the importance of improved affect detection models over various affect feature extraction techniques that was proposed in chapter 3 for model EMoRec.

8 Conclusions and Future Directions

The analysis of affect, feelings, emotions, sentiments, and opinions in text data such as reviews, news articles, tweets, including sentiment analysis, emotion detection and sarcasm detection, is considered affect detection. There has been a growing interest in developing the computational methods for affect detection from text in recent years. Although affective analysis is quite challenging for all the tasks, there are great opportunities from the natural language processing point of view to address these problems. In addition, recommendation systems have become pervasive over the last decade, providing users with personalized search results, music suggestions, news articles to read, and shopping hints. On the other hand, human affects are widely regarded as important predictors of behaviour and preference, which are crucial factors in decision making based on cognitive psychology[103]. Despite the impressive achievements in affect detection approaches and recommendation models solely in many domains, there is a significant gap between considering different affect information in recommendation systems and affective analysis techniques. Throughout this dissertation, we enriched word representation learning

centred around two particular modules - affect detection and recommendation systems. In a nutshell, we took the first steps towards bridging the gap between needs in personalized recommendation systems and capacities of affective analysis approaches to address these demands through four tasks. In particular, we attempted to indicate that improving the affect detection approaches can improve the recommendation process.

8.1 Summary of Approaches and Contributions

In this dissertation, we framed the main challenges of affect detection in natural language processing into feasible problems in recommendation systems. First, we argued that affective information plays a crucial role in decision making and predictive models of recommendation systems. As such, it is of paramount importance to understand the potential of different affective factors in the user/item context, which makes the prediction more accurate. Given this fact, we designed a framework to evaluate the importance of various emotion-based features in recommendation models. The summary of contributions are as follows:

- We extracted the most relevant emotion-based features associated with both items and users using various affect detection approaches for use in recommendation models in the Music and News domains.

- We devised a number of state-of-the-art models for generating recommendations to

incorporate the additional affect features and proposed an ensemble model EMoRec by combining three models (*Boost Blend + Deep MF + DNN Ensemble*), which significantly outperformed other state-of-the-art recommendation methods.

Moreover, inspired by the latest advances in enriching word representation learning approaches for affective tasks, we considered studying the role of applying various pre-processing factors applied to word representation learning in different stages for affect detection. The key contributions are:

- We conducted a comprehensive analysis of the role of various pre-processing techniques in affective tasks (including sentiment analysis, emotion classification and sarcasm detection), employing different word embedding models, which indicated that negation handling and part-of-speech tagging to be more beneficial factors in affective analysis tasks.

- We performed a comparative analysis of the accuracy performance of word vector models when pre-processing is applied to large training corpora *before* learning word embeddings at the training phase (training data) and/or *later* at the downstream task phase (classification dataset). Interestingly, we obtained the best results when pre-processing factors are applied only to the training corpus or when it is applied to both the training corpus and the classification dataset of interest.

Motivated by the observation that various pre-processing factors applied to word

embeddings can significantly improve the performance of different affective tasks, we investigated whether customized combinations of pre-processing factors at different stages of word representation learning in affective tasks can improve the performance. The contributions are summarized as follows:

- We proposed different combinations of pre-processing factors that are most suitable for each affective task, employing seven embedding models which demonstrated appropriate combination of pre-processing techniques on the embedding-training corpora and on classification datasets better improved the performance rather than the general combination of pre-processing techniques.

- We conducted experiments to study the role of each combination of pre-processing factors for a specific affective task and indicated that applying different sets of pre-processing factors which are more suitable for each downstream task has significant effects on the performance when it is applied to both the embedding-training corpus and the classification dataset of interest.

In line with our research goals, we designed two novel deep neural network language models for affect detection while each model extends the architecture of BERT, which to the best of our knowledge, was the first attempt to directly extend BERT's architecture (rather than using the already pre-trained BERT embeddings). The proposed models effectively incorporated both affective and contextual features of text to build a classifier.

The summary of contributions is as follows:

- We proposed two novel deep neural network language models (ACE 1 and ACE 2) by incorporating affective and contextual features for affect detection, which demonstrated that they significantly outperform current state-of-the-art models. The proposed models learned the affective representation of a document, using a Bi-LSTM architecture with the multi-head attention mechanism.

- We presented the effects of the proposed affective and contextual model on three affective tasks, including sarcasm detection, emotion detection and sentiment analysis. The results indicated that binary classifications (e.g. sarcasm detection and sentiment analysis) could benefit more from the models than multi-class classification (e.g. emotion detection).

Finally, we evaluated the usefulness of our proposed affect detection approaches in recommendation systems, which indicated that improved affective detection techniques are far more beneficial than using various affective feature extraction techniques in recommendation models. The summary of the key contributions is as follows:

- We applied the proposed customized pre-processing factors to the word representation learning stage of ACE 1, and then the generated embeddings from ACE 1 was used in ACE 2 for the affective feature extraction in different recommendation

algorithms. Our analysis revealed that adding affective features from ACE 1 and ACE 2 improved the predictive performance for all the recommendation methods.

- We designed an Affect-Aware Recommendation (AARec) model by extending the state-of-the-art attention-based recommendation algorithm by considering the affective and contextual embedding representation in the recommendation process. The results demonstrated that affective and contextual feature embeddings produced by ACE 1 and ACE 2 along with pre-processing techniques can significantly improve the prediction task in comparison to affective feature extraction proposed in EMoRec.

Last but not least, the outcomes of this study have been published in the top tier conference proceedings in the field of natural language processing, such as the proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020) [12], proceedings of the 28th International Conference on Computational Linguistics (COLING 2020) [14] as well as proceedings of the 7th International Workshop on News Recommendation and Analytics (INRA 2019) in conjunction with the 13th ACM Conference on Recommender Systems (RecSys 2019) [13] .

8.2 Future Directions

As this work is among the early stages towards considering implicit affect information in recommendations by improving the affect detection techniques, there are a lot of interesting directions for future work.

8.2.1 Recommendation with Affective Information Through Other Cues

Employing affective context in recommendations appears to be a promising direction of research. While the scope of our current study was limited to affects extracted either using various affect detection approaches (Chapter 3) or improving the affect detection by enriching word representation learning (Chapter 7) from textual information, there is evidence that different affects can be extracted through other means of communication, such as *audio* and *video*, or other cues [165]. Hence, an interesting direction of affect-aware recommendations can be to explore in-session recommendations using affects information in other cues. It is becoming evident from our data analysis that a user's mood can fluctuate over time (e.g. Figure 3.1). However, while these fluctuations can be significant over time, they are probably more stable within a limited time, such as a single session (i.e., a session of continuously listening to songs, or a session of continuously navigating and reading articles). Therefore, it seems reasonable to design affect-aware recommendation algorithms that weight more (i.e., promote) items that match the affective

context of a user's current session and weight less or filter out (i.e., demote) items that conflict with it from cues other than text such as audio or video.

8.2.2 Negation Scope and Negation Handling

In Chapters 4 and 5, we described the importance of different pre-processing factors for word representation learning in affective tasks while indicating negation handling (*neg*) as the most effective pre-processing in affective tasks. However, negation identification and detecting its scope within a sentence in biomedical abstracts or clinical notes can be beneficial for other domains such as health and medical systems. There is a wealth of clinical information in digital health records that can theoretically be used for a number of clinical tasks [47]. This information includes the absence or existence of medical conditions. For example, the sentence " Patient had not eaten for the past three days, felt nausea, and then collapsed" contains three medically relevant "event" that might be used as an input to a clinical decision support system such as eating behaviour, feeling nausea and loss of consciousness, while only one of them (e.g. normal eating behaviour) is negated. Hence, it is crucial to distinguish between the two since the negated events and the non-negated events often have very different prognostic value. Moreover, the majority of state-of-the-art negation scope detection systems rely on the availability of the cue information such as *no*, *not*, and *should not*, which limits the applicability of the model to non-cue annotated

[47] http://biostat.mc.vanderbilt.edu/wiki/Main/DataSets

data. Therefore, designing a model to detect negation scope by utilizing the syntactic structure of a sentence (e.g. parse trees) [27, 164], which is not directly dependent on the negation cues, can be an interesting future work.

8.2.3 Multilingual Model

The proposed models in this dissertation were evaluated only on English datasets. However, most of the available models, such as BERT and ELMo are language-agnostic and designed for other languages and domains. For example, the affective and contextual model (ACE 1 from Chapter 6) could be generated from source corpus originating in another language. With recent easy availability of numerous multilingual datasets [60, 72, 114], as future work, it would be interesting to experiment with data from languages other than English.

8.2.4 Learning of Affective Representations Through Graphs

Several ways of further enhancing the word representation learning, specifically word embeddings in affective tasks, were proposed in this dissertation (Chapter 4 and 5). The proposed models mostly focused on learning affective and contextual representations at the word or document level (Chapter 6 and Chapter 7). However, affect in text can be conveyed at the phrase level too (e.g. little sadness, amazingly beautiful). Furthermore, with the surge of approaches that seek to learn representations that encode structural information

in graphs [182], some interesting lines of future work could include considering these alternate models for learning affective representations through the graph structure. One interesting future work can be incorporating affective, semantic and syntactic information in word representation learning at the phrase level using graph neural networks.

8.2.5 Integrating the Proposed Models into One System

We argued that relevant affective information does matter in recommendation systems and that it is important to take this contextual information into account when providing recommendations (Chapter 3 and 7). We also explained that the affective information can be extracted in different ways, such as various techniques that were explained in Chapter 3 for a recommendation process. We have also shown various affective detection approaches, including pre-processing/customized pre-processing techniques and affective-contextual models (ACE 1 and ACE 2) to detect different types of affects in text. An interesting future work could be building a model that combines these proposed models into a single recommendation system, which can be more cost and time efficient than the current AARec model presented in Chapter 7 that has to exploit each proposed models' results solely and then combine them later.

Bibliography

[1] B. Agarwal, N. Mittal, P. Bansal, and S. Garg. Sentiment analysis using common-sense and context information. *Computational Intelligence and Neuroscience*, 2015.

[2] A. Agrawal and A. An. Unsupervised Emotion Detection from Text Using Semantic and Syntactic Relations. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 346–353, Dec. 2012.

[3] A. Agrawal and A. An. Affective Representations for Sarcasm Detection. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1029–1032, Ann Arbor MI USA, June 2018. ACM.

[4] A. Agrawal, A. An, and M. Papagelis. Learning emotion-enriched word representations. In *Proceedings of the 27th International Conference on Computational*

Linguistics, pages 950–961, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics.

[5] A. Agrawal, A. An, and M. Papagelis. Leveraging transitions of emotions for sarcasm detection. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1505–1508, 2020.

[6] D. Al-Ghadhban, E. Alnkhilan, L. Tatwany, and M. Al-Razgan. Arabic sarcasm detection in twitter. *2017 International Conference on Engineering and MIS (ICEMIS)*, pages 1–7, 2017.

[7] M. Al Masum Shaikh, H. Prendinger, and M. Ishizuka. Emotion sensitive news agent (esna): A system for user centric emotion sensing from the news. *Web Intelligence and Agent Systems*, 8(4):377–396, 2010.

[8] C. O. Alm, D. Roth, and R. Sproat. Emotions from text: Machine learning for text-based emotion prediction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 579–586, Vancouver, British Columbia, Canada, Oct. 2005. Association for Computational Linguistics.

[9] S. Aman and S. Szpakowicz. Identifying expressions of emotion in text. In *Text, speech and dialogue*, pages 196–205. Springer, 2007.

[10] S. Amir, B. C. Wallace, H. Lyu, P. Carvalho, and M. J. Silva. Modelling context with user embeddings for sarcasm detection in social media. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 167–177, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

[11] G. Angiani, L. Ferrari, T. Fontanini, P. Fornacciari, E. Iotti, F. Magliani, and S. Manicardi. A comparison between preprocessing techniques for sentiment analysis in twitter. In *Proceedings of the 2nd International Workshop on Knowledge Discovery on the WEB, KDWeb*, 2016.

[12] N. Babanejad, A. Agrawal, A. An, and M. Papagelis. A comprehensive analysis of preprocessing for word representation learning in affective tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5799–5810, Online, July 2020. Association for Computational Linguistics.

[13] N. Babanejad, A. Agrawal, H. Davoudi, A. An, and M. Papagelis. Leveraging emotion features in news recommendations. In *Proceedings of the 7'th International Workshop on News Recommendation and Analytics (INRA'19) in conjunction with RecSys'19, Copenhagen, Denmark, September 16 - 20, 2019.*, 2019.

[14] N. Babanejad, H. Davoudi, A. An, and M. Papagelis. Affective and contextual embedding for sarcasm detection. In *Proceedings of the 28th International Con-*

ference on Computational Linguistics, pages 225–243, Barcelona, Spain (Online), Dec. 2020. International Committee on Computational Linguistics.

[15] S. Baccianella, A. Esuli, and F. Sebastiani. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *Lrec*, volume 10, pages 2200–2204, 2010.

[16] R. Bagozzi, M. Gopinath, and P. Nyer. The role of emotions in marketing. *Journal of the Academy of Marketing Science*, 27:184–206, 04 1999.

[17] B. Bai and Y. Fan. Incorporating Field-aware Deep Embedding Networks and Gradient Boosting Decision Trees for Music Recommendation. In *The 11th ACM International Conference on Web Search and Data Mining(WSDM)*, page 7, London, England, 2017. ACM.

[18] F. Benamara, B. Chardon, Y. Mathieu, V. Popescu, and N. Asher. How do negation and modality impact on opinions? In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*, ExProM '12, pages 10–18, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[19] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, Mar. 2003.

[20] E. Boiy, P. Hens, K. Deschacht, and M.-F. Moens. Automatic sentiment analysis in on-line text. In *Proceedings of the 11th International Conference on Electronic Publishing ELPUB2007*, 2007.

[21] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, Dec 2017.

[22] M. Bouazizi and T. O. Ohtsuki. A Pattern-Based Approach for Sarcasm Detection on Twitter. *IEEE Access*, 4:5477–5488, 2016.

[23] S. Brave and C. Nass. Emotion in human–computer interaction. *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*, 01 2002.

[24] J. Brooke. A semantic approach to automated text sentiment analysis. Master's thesis, SIMON FRASER UNIVERSITY, British Columbia, BC, Canada, 2009.

[25] C. Burgers, M. v. Mulken, and P. J. Schellens. Verbal Irony: Differences in Usage Across Written Genres. *Journal of Language and Social Psychology*, 31(3):290–310, 2012. _eprint: https://doi.org/10.1177/0261927X12446596.

[26] J. Camacho-Collados and M. T. Pilehvar. On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment

analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics, 2018.

[27] J. Carrillo-de Albornoz and L. Plaza. An emotion-based model of negation, intensifiers, and modality for polarity and intensity classification. *Journal of the American Society for Information Science and Technology*, 64:1618–1633, 08 2013.

[28] P. Carvalho, L. Sarmento, M. J. Silva, and E. de Oliveira. Clues for detecting irony in user-generated contents: Oh...!! it's "so easy"FIX ME!!!!;-). In *Proceedings of the 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion*, TSA '09, page 53–56, New York, NY, USA, 2009. Association for Computing Machinery.

[29] P. Carvalho, L. Sarmento, J. Teixeira, and M. J. Silva. Liars and saviors in a sentiment annotated corpus of comments to political debates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 564–568, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[30] S. Castro, D. Hazarika, V. Pérez-Rosas, R. Zimmermann, R. Mihalcea, and S. Poria. Towards multimodal sarcasm detection (an-obviously-perfect paper). *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[31] E. Cecilia and A. Ovesdotter. *Affect in text and speech*. ProQuest, Citeseer, 2008.

[32] S. Channon, A. Pellijeff, and A. Rule. Social cognition after head injury: Sarcasm and theory of mind. *Brain and Language*, 93(2):123–134, May 2005.

[33] C.-M. Chen, M.-F. Tsai, J.-Y. Liu, and Y.-H. Yang. Using emotional context from article for contextual music recommendation. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, page 649–652, New York, NY, USA, 2013. Association for Computing Machinery.

[34] L. Chen, G. Chen, and F. Wang. Recommender systems based on user reviews: The state of the art. *User Modeling and User-Adapted Interaction*, 25(2):99–154, June 2015.

[35] T. Chen and C. Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. event-place: San Francisco, California, USA.

[36] X. V. Chen and T. Y. Tang. Combining content and sentiment analysis on lyrics for a lightweight emotion-aware chinese song recommendation system. In *Proceedings of the 2018 10th International Conference on Machine Learning and Computing*, pages 85–89. ACM, 2018.

[37] B. Chiu, G. Crichton, A. Korhonen, and S. Pyysalo. How to train good word embeddings for biomedical nlp. In *Proceedings of the 15th workshop on biomedical natural language processing*, pages 166–174, 2016.

[38] J. Coates and D. Bollegala. Frustratingly easy meta-embedding – computing meta-embeddings by averaging source word embeddings. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018.

[39] T. Danisman and A. Alpkocak. Feeler: Emotion classification of text using vector space model. In *Proceedings of the AISB 2008 Symposium on Affective Language in Human and Machine, AISB 2008 Convention Communication, Interaction and Social Intelligence*, volume 1, page 53, 2008.

[40] D. Davidov, O. Tsur, and A. Rappoport. Semi-supervised recognition of sarcasm in twitter and Amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 107–116, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[41] T. Davidson, D. Warmsley, M. Macy, and I. Weber. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515, 2017.

[42] H. Davoudi, A. An, M. Zihayat, and G. Edall. Adaptive paywall mechanism for digital news media. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 205–214, New York, NY, USA, 2018. ACM.

[43] H. Davoudi, M. Zihayat, and A. An. Time-Aware Subscription Prediction Model for User Acquisition in Digital News Media. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, Proceedings, pages 135–143. Society for Industrial and Applied Mathematics, June 2017.

[44] L. De Bruyne, O. De Clercq, and V. Hoste. LT3 at SemEval-2018 task 1: A classifier chain to detect emotions in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 123–127, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[45] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[46] N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, and N. Bhamidipati. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, page 29–30, New York, NY, USA, 2015. Association for Computing Machinery.

[47] A. V. Dorogush, V. Ershov, and A. Gulin. CatBoost: gradient boosting with categorical features support. *Mathematics, Computer Science*, Oct. 2018.

[48] J. Druckman and R. McDermott. Emotion and the framing of risky choice. *Political Behavior*, 30(3):297–321, Sept. 2008.

[49] C. Du, H. Sun, J. Wang, Q. Qi, and J. Liao. Adversarial and domain-aware BERT for cross-domain sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4019–4028, Online, July 2020. Association for Computational Linguistics.

[50] P. Ekman. Expression and the nature of emotion. *Approaches to emotion*, 3:19–344, 1984.

[51] P. Ekman. An Argument For Basic Emotions. *Cognition & Emotion*, 6:169–200, May 1992.

[52] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*, 2014.

[53] 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 International Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[54] J. Feng, Y. Yu, and Z.-H. Zhou. Multi-Layered Gradient Boosting Decision Trees. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3551–3561. Curran Associates, Inc., 2018.

[55] G. Forman. An extensive empirical study of feature selection metrics for text classification [j]. *Journal of Machine Learning Research - JMLR*, 3, 03 2003.

[56] E. Forslid and N. Wikén. *Automatic irony-and sarcasm detection in Social media*. PhD thesis, UPPSALA UNIVERSITET, 2015.

[57] B. Fortuna, C. Fortuna, and D. Mladenić. Real-time news recommender system. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 583–586. Springer, 2010.

[58] D. Ghazi, D. Inkpen, and S. Szpakowicz. Prior and contextual emotion of words in sentential context. *Computer Speech & Language*, 28(1):76–92, Jan. 2014.

[59] D. Ghosh, A. Richard Fabbri, and S. Muresan. The Role of Conversation Context for Sarcasm Detection in Online Interactions. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 186–196, Saarbrücken, Germany, 2017. Association for Computational Linguistics.

[60] M. Giatsoglou, M. G. Vozalis, K. Diamantaras, A. Vakali, G. Sarigiannidis, and

K. C. Chatzisavvas. Sentiment analysis leveraging emotions and word embeddings. *Expert Systems with Applications*, 69:214 – 224, 2017.

[61] C. Goddard. Interjections and emotion (with special reference to "surprise" and "disgust"). *Emotion Review*, 6:53 – 63, 2014.

[62] G. Gonzalez, J. L. de la Rosa, M. Montaner, and S. Delfin. Embedding emotional context in recommender systems. In *Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop*, ICDEW '07, pages 845–852, Washington, DC, USA, 2007. IEEE Computer Society.

[63] R. González-Ibáñez, S. Muresan, and N. Wacholder. Identifying sarcasm in twitter: A closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 581–586, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[64] D. Gossi and M. H. Gunes. Lyric-based music recommendation. In *Complex Networks VII*, pages 301–310. Springer, 2016.

[65] V. Gratian and M. Haid. Braint at iest 2018: Fine-tuning multiclass perceptron for implicit emotion classification. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 243–247, 2018.

[66] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *NEURAL NETWORKS*, pages 5–6, 2005.

[67] X. Guo and J. Li. A novel twitter sentiment analysis model with baseline correlation for financial market prediction with improved efficiency. In *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 472–477, 2019.

[68] L. A. Gutnik, A. F. Hakimzada, N. A. Yoskowitz, and V. L. Patel. The role of emotion in decision-making: A cognitive neuroeconomic approach towards understanding sexual risk behavior. *Journal of Biomedical Informatics*, 39(6):720 – 736, 2006.

[69] E. Haddi, X. Liu, and Y. Shi. The role of text pre-processing in sentiment analysis. *Procedia Computer Science*, 17:26–32, 12 2013.

[70] B. Han, S. Rho, S. Jun, and E. Hwang. Music emotion classification and context-based music recommendation. *Multimedia Tools and Applications*, 47(3):433–460, May 2010.

[71] B.-J. Han, S. Rho, S. Jun, and E. Hwang. Music emotion classification and context-based music recommendation. *Multimedia Tools and Applications*, 47(3):433–460, 2010.

[72] K. Hashimoto, R. Buschiazzo, J. Bradbury, T. Marshall, R. Socher, and C. Xiong. A high-quality multilingual dataset for structured documentation translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 116–127, Florence, Italy, Aug. 2019. Association for Computational Linguistics.

[73] K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1923–1933, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.

[74] D. Hazarika, S. Poria, S. Gorantla, E. Cambria, R. Zimmermann, and R. Mihalcea. CASCADE: Contextual sarcasm detection in online discussion forums. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1837–1848, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics.

[75] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web - WWW '17*, pages 173–182, Perth, Australia, 2017. ACM Press.

[76] D. I. Hernández Farías, M. Montes-y Gómez, H. J. Escalante, P. Rosso, and

V. Patti. A knowledge-based weighted KNN for detecting Irony in Twitter. In *Proceedings17th Mexican International Conference on Artificial Intelligence, MICAI 2018*, volume 11289, pages 194–206. Springer Verlag, 2018. Accepted: 2019-04-14T18:17:25Z.

[77] D. Hershcovich, A. Toledo, A. Halfon, and N. Slonim. Syntactic interchangeability in word embedding models. *arXiv preprint arXiv:1904.00669*, 2019.

[78] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. In *Proceedings of the International Conference on Learning Representations.*, volume abs/1511.06939, 2016.

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

[80] Y. Hu. *A Model-Based Music Recommendation System for Individual Users and Implicit User Groups*. PhD thesis, University of Miami, 2014.

[81] Y. Hu, X. Chen, and D. Yang. Lyric-based song emotion detection with affective lexicon and fuzzy clustering method. In *Proceedings of ISMIR 2009*, pages 123–128, 2009.

[82] Y. Hu, D. Li, and O. Mitsunori. EVALUATION ON FEATURE IMPORTANCE

FOR FAVORITE SONG DETECTION. In *14th International Society for Music Information Retrieval Conference(ISMIR)*, page 6, Brazil, 2013. (ISMIR).

[83] L. Huang, F. Gino, and A. D. Galinsky. The highest form of intelligence: Sarcasm increases creativity for both expressers and recipients. *Organizational Behavior and Human Decision Processes*, 131(C):162–177, 2015.

[84] S. Ilić, E. Marrese-Taylor, J. Balazs, and Y. Matsuo. Deep contextualized word representations for detecting sarcasm and irony. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 2–7, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics.

[85] Z. Jianqiang and G. Xiaolin. Comparison research on text pre-processing methods on twitter sentiment analysis. *IEEE Access*, 5:2870–2879, 2017.

[86] A. Joshi, V. Sharma, and P. Bhattacharyya. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 757–762, Beijing, China, July 2015. Association for Computational Linguistics.

[87] A. Joshi, V. Tripathi, P. Bhattacharyya, and M. J. Carman. Harnessing sequence labeling for sarcasm detection in dialogue from TV series 'Friends'. In *Proceedings*

of The 20th SIGNLL Conference on Computational Natural Language Learning, pages 146–155, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

[88] V. Jovanović. The form, position and meaning of interjections in english. *Facta Universitatis: Series Linguistics and Litearture*, 06 2004.

[89] M. Karimi, D. Jannach, and M. Jugovac. News recommender systems – Survey and roads ahead. *Information Processing & Management*, 54(6):1203–1227, Nov. 2018.

[90] A. Khade, A. Prof, and D. M. Modeling customer behavior for efficient recommendation systems. *SSRN Electronic Journal*, 7, 01 2020.

[91] G. Khanvilkar and D. Vora. Sentiment analysis for product recommendation using random forest. *International Journal of Engineering and Technology(UAE)*, 7:87–89, 06 2018.

[92] D. Khattar, V. Kumar, M. Gupta, and V. Varma. Neural Content-Collaborative Filtering for News Recommendation. In *NewsIR'18 Workshop*, page 6, Grenoble, France, Mar. 2018.

[93] M. Khodak, N. Saunshi, and K. Vodrahalli. A large self-annotated corpus for sarcasm. *arXiv preprint arXiv:1704.05579*, 2017.

[94] D. Kiela, C. Wang, and K. Cho. Dynamic meta-embeddings for improved sentence representations. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[95] Y. Kim, H. Lee, and K. Jung. AttnConvnet at SemEval-2018 task 1: Attention-based convolutional neural networks for multi-label emotion classification. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 141–145, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[96] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[97] B. Krishnamurthy, N. Puri, and R. Goel. Learning vector-space representations of items for recommendations using word embedding models. *Procedia Computer Science*, 80:2205 – 2210, 2016. International Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA.

[98] M. Kuhn and K. Johnson. *Feature Engineering and Selection: A Practical Approach for Predictive Models*. Chapman & Hall/CRC Data Science Series. CRC Press, 2019.

[99] A. Kumar, V. T. Narapareddy, V. Aditya Srikanth, A. Malapati, and L. B. M. Neti.

Sarcasm Detection Using Multi-Head Attention Based Bidirectional LSTM. *IEEE Access*, 8:6388–6397, 2020. Conference Name: IEEE Access.

[100] H. J. Lee and S. J. Park. Moners: A news recommender for the mobile web. *Expert Systems with Applications*, 32(1):143–150, 2007.

[101] E. L. Lehmann. *Introduction to Neyman and Pearson (1933) On the Problem of the Most Efficient Tests of Statistical Hypotheses*, pages 67–72. Springer New York, New York, NY, 1992.

[102] J. S. Lerner, Y. Li, P. Valdesolo, and K. S. Kassam. Emotion and decision making. *Annual Review of Psychology*, 66(1):799–823, 2015. PMID: 25251484.

[103] J. S. Lerner, Y. Li, P. Valdesolo, and K. S. Kassam. Emotion and decision making. *Annual Review of Psychology*, 66(1):799–823, 2015. PMID: 25251484.

[104] O. Levy and Y. Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 302–308, 2014.

[105] O. Levy, Y. Goldberg, and I. Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

[106] P. Lison and A. Kutuzov. Redefining context windows for word embedding models:

An experimental study. In *Proceedings of the 21st Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 284–288, Gothenburg, Sweden, May 2017. Association for Computational Linguistics.

[107] 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, 2019.

[108] E. Lunando and A. Purwarianti. Indonesian social media sentiment analysis with sarcasm detection. *2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, September 2013.

[109] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[110] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, United States, 2009.

[111] M. Y. Manohar and P. Kulkarni. Improvement sarcasm analysis using nlp and corpus based approach. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 618–622, 2017.

[112] C. May, A. Wang, S. Bordia, S. R. Bowman, and R. Rudinger. On measuring social biases in sentence encoders. *Proceedings of the 2019 Conference of the North*, 2019.

[113] O. Melamud, D. McClosky, S. Patwardhan, and M. Bansal. The role of context types and dimensionality in learning word embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1030–1040, San Diego, California, June 2016. Association for Computational Linguistics.

[114] P. Michel and G. Neubig. MTNT: A testbed for machine translation of noisy text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 543–553, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics.

[115] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[116] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. *arXiv:1310.4546 [cs, stat]*, Oct. 2013. arXiv: 1310.4546.

[117] G. A. Miller. Wordnet: A lexical database for english. *Association for Computing Machinery, Commun. ACM*, 38(11):39–41, Nov. 1995.

[118] A. Mishra, D. Kanojia, S. Nagar, K. Dey, and P. Bhattacharyya. Harnessing cognitive features for sarcasm detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1095–1104, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

[119] R. Misra and P. Arora. Sarcasm detection using hybrid neural network. *arXiv preprint arXiv:1908.07414*, 2019.

[120] J. Mizgajski and M. Morzy. Affective recommender systems in online news industry: how emotions influence reading choices. *User Modeling and User-Adapted Interaction*, 29:345–379, 2018.

[121] S. M. Mohammad. Word affect intensities. *ArXiv*, abs/1704.08798, 2017.

[122] S. M. Mohammad and F. Bravo-Marquez. WASSA-2017 Shared Task on Emotion Intensity. In *EMNLP 2017 Workshop on Computational Approaches to Subjectivity, Sentiment, and Social Media (WASSA)*, Proceedings, Copenhagen, Denmark, Aug. 2017. EMNLP.

[123] S. M. Mohammad, P. Sobhani, and S. Kiritchenko. Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT)*, 17(3):26, 2017.

[124] S. M. Mohammad and P. D. Turney. Crowdsourcing a Word–Emotion Association Lexicon. *Computational Intelligence*, 29(3):436–465, Aug. 2013.

[125] A. Montes-García, J. M. Álvarez-Rodríguez, J. E. Labra-Gayo, and M. Martínez-Merino. Towards a journalist-based news recommendation system: The wesomender approach. *Expert Systems with Applications*, 40(17):6735–6741, 2013.

[126] C. E. Moody. Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec. *arXiv:1605.02019 [cs]*, May 2016. arXiv: 1605.02019.

[127] Y. Moshfeghi, B. Piwowarski, and J. M. Jose. Handling data sparsity in collaborative filtering using emotion and semantic based features. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 625–634. ACM, 2011.

[128] M. Mozafari, R. Farahbakhsh, and N. Crespi. A bert-based transfer learning approach for hate speech detection in online social media. *Studies in Computational Intelligence*, page 928–940, Nov 2019.

[129] H. Mulki, C. B. Ali, H. Haddad, and I. Babaoğlu. Tw-star at semeval-2018 task 1: Preprocessing impact on multi-label emotion classification. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 167–171, 2018.

[130] P. Nakov, A. Ritter, S. Rosenthal, V. Stoyanov, and F. Sebastiani. SemEval-2016

task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, June 2016. Association for Computational Linguistics.

[131] A. Odić, M. Tkalčič, J. F. Tasič, and A. Košir. Predicting and detecting the relevant contextual information in a movie-recommender system. *Interacting with Computers*, 25(1):74–90, 2013.

[132] S. Oraby, V. Harrison, L. Reed, E. Hernandez, E. Riloff, and M. Walker. Creating and characterizing a diverse corpus of sarcasm in dialogue. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 31–41, Los Angeles, Sept. 2016. Association for Computational Linguistics.

[133] S. O. Orimaye, S. M. Alhashmi, and S. Eu-gene. Sentiment Analysis Amidst Ambiguities in Youtube Comments on Yoruba Language (Nollywood) Movies. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12 Companion, pages 583–584, New York, NY, USA, 2012. ACM.

[134] Y. Ozaki, M. Yano, and M. Onishi. Effective hyperparameter optimization using Nelder-Mead method in deep learning. *IPSJ Transactions on Computer Vision and Applications*, 9(1):20, Nov. 2017.

[135] M. Pantic and A. Vinciarelli. Implicit human-centered tagging [social sciences]. *IEEE Signal Processing Magazine*, 26(6):173–180, 2009.

[136] M. Papagelis and D. Plexousakis. Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents. *Engineering Applications of Artificial Intelligence*, 18(7):781–789, 2005.

[137] M. Papagelis, D. Plexousakis, and T. Kutsuras. Alleviating the sparsity problem of collaborative filtering using trust inferences. In *International Conference on Trust Management*, pages 224–239. Springer, 2005.

[138] A. H. Parizi and M. Kazemifard. Emotional news recommender system. In *2015 Sixth International Conference of Cognitive Science (ICCS)*, pages 37–41. IEEE, 2015.

[139] J. H. Park, P. Xu, and P. Fung. PlusEmo2Vec at SemEval-2018 task 1: Exploiting emotion knowledge from emoji and #hashtags. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 264–272, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[140] C. G. Patil and S. Patil. Use of porter stemming algorithm and svm for emotion extraction from news headlines. In *International Journal of Electronics, Communication and Soft Computing Science and Engineering*, 2013.

[141] S. Pecar, M. Farkas, M. Simko, P. Lacko, and M. Bielikova. Nl-fiit at iest-2018: Emotion recognition utilizing neural networks and multi-level preprocessing. In

Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pages 217–223, 2018.

[142] J. Pennington, R. Socher, and C. 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.

[143] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018.

[144] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *ArXiv*, abs/1802.05365, 2018.

[145] M. E. Peters, M. Neumann, R. Logan, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith. Knowledge enhanced contextual word representations. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[146] R. W. Picard, E. Vyzas, and J. Healey. Toward machine emotional intelligence: analysis of affective physiological state. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1175–1191, 2001.

[147] B. Pickering, D. Thompson, and R. Filik. Examining the emotional impact of sarcasm using a virtual environment. *Metaphor and Symbol*, 33(3):185–197, July 2018. Publisher: Routledge _eprint: https://doi.org/10.1080/10926488.2018.1481261.

[148] S. Poria, E. Cambria, D. Hazarika, and P. Vij. A deeper look into sarcastic tweets using deep convolutional neural networks. *ArXiv*, abs/1610.08815, 2016.

[149] R. A. Potamias, G. Siolas, and A. G. Stafylopatis. A transformer-based approach to irony and sarcasm detection, 2019.

[150] T. Ptácek, I. Habernal, and J. Hong. Sarcasm detection on czech and english twitter. In *COLING*, 2014.

[151] A. Rajadesingan, R. Zafarani, and H. Liu. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, page 97–106, New York, NY, USA, 2015. Association for Computing Machinery.

[152] R. Rakov and A. Rosenberg. "sure, i did the right thing": a system for sarcasm detection in speech. In *INTERSPEECH*, 2013.

[153] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[154] E. Riloff, A. Qadir, P. Surve, L. De Silva, N. Gilbert, and R. Huang. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 704–714, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics.

[155] S. L. Rose, R. Venkatesan, G. Pasupathy, and P. Swaradh. A lexicon-based term weighting scheme for emotion identification of tweets. *International Journal of Data Analysis Techniques and Strategies*, 10(4):369–380, 2018.

[156] M. Rumiantcev. *Music adviser : emotion-driven music recommendation ecosystem*. PhD thesis, Department of Mathematical Information Technology Oleksiy Khriyenko, 2017.

[157] A. D. S, R. S, S. M. Rajendram, and M. T T. SSN MLRG1 at SemEval-2018 task 1: Emotion and sentiment intensity detection using rule based feature selection. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 324–328, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[158] H. Saif, M. Fernandez, Y. He, and H. Alani. On stopwords, filtering and data sparsity for sentiment analysis of twitter. In *Proceedings of the Ninth International*

Conference on Language Resources and Evaluation (LREC'14), pages 810–817, Reykjavik, Iceland, may 2014. European Language Resources Association (ELRA).

[159] H. Schuff, J. Barnes, J. Mohme, S. Padó, and R. Klinger. Annotation, modelling and analysis of fine-grained emotions on a stance and sentiment detection corpus. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 13–23, 2017.

[160] D. Seal, U. K. Roy, and R. Basak. Sentence-level emotion detection from text based on semantic rules. In *Information and Communication Technology for Sustainable Development*, pages 423–430. Springer, 2020.

[161] N. Shahbazi, M. Chahhou, and J. Gryz. Truncated SVD-based Feature Engineering for Music Recommendation. In *The 11th ACM International Conference on Web Search and Data Mining(WSDM)*, page 7, London, England, 2017. The 11th ACM International Conference on Web Search and Data Mining.

[162] B. Shiv and A. Fedorikhin. Heart and Mind in Conflict: The Interplay of Affect and Cognition in Consumer Decision Making. *Journal of Consumer Research*, 26(3):278–292, 12 1999.

[163] S. N. Shivhare and S. K. Saritha. Emotion Detection From Text Documents. *International Journal of Data Mining & Knowledge Management Process*, 4(6):51–57, Nov. 2014.

[164] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria, Aug. 2013. Association for Computational Linguistics.

[165] M. Soleymani, S. Asghari-Esfeden, Y. Fu, and M. Pantic. Analysis of eeg signals and facial expressions for continuous emotion detection. *IEEE Transactions on Affective Computing*, 7(1):17–28, 2016.

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

[167] C. Strapparava, A. Valitutti, and others. WordNet Affect: an Affective Extension of WordNet. In *LREC*, volume 4, pages 1083–1086, Lisbon, Portugal, 2004. European Language Resources Association (ELRA).

[168] F. Strohm. The impact of intensifiers, diminishers and negations on emotion expressions. B.S. thesis, University of Stuttgart, 2017.

[169] C. Sun, L. Huang, and X. Qiu. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics:*

Human Language Technologies, Volume 1 (Long and Short Papers), pages 380–385, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[170] S. Symeonidis, D. Effrosynidis, and A. Arampatzis. A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis. *Expert Systems with Applications*, 110:298–310, 2018.

[171] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Investigation of various matrix factorization methods for large recommender systems. In *2008 IEEE International Conference on Data Mining Workshops*, pages 553–562. IEEE, 2008.

[172] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[173] J. Tang and K. Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM)*, 2018.

[174] Y. Tay, A. T. Luu, S. C. Hui, and J. Su. Reasoning with sarcasm by reading in-between. *Proceedings of 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

[175] M. Tkalcic, A. Kosir, J. Tasivc, and M. Kunaver. Affective recommender systems: the role of emotions in recommender systems. In *5th ACM Conference on Recommender Systems (RecSys*, pages 9–13, 01 2011.

[176] M. Tkalčič, U. Burnik, A. Odić, A. Košir, and J. Tasič. Emotion-Aware Recommender Systems – A Framework and a Case Study. In S. Markovski and M. Gusev, editors, *ICT Innovations 2012*, pages 141–150, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[177] O. Tsur, D. Davidov, and A. Rappoport. ICWSM-A Great Catchy Name: Semi-Supervised Recognition of Sarcastic Sentences in Online Product Reviews. In *In Fourth International AAAI Conference on Weblogs and Social Media*, 2010.

[178] P. Uhr, J. Zenkert, and M. Fathi. Sentiment analysis in financial markets - a framework to utilize the human ability of word association for analyzing stock market news reports. In *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, volume 2014, 10 2014.

[179] A. K. Uysal and S. Gunal. The impact of preprocessing on text classification. *Information Processing and Management*, 50:104 – 112, 01 2014.

[180] C. Van Hee, E. Lefever, and V. Hoste. SemEval-2018 task 3: Irony detection in English tweets. In *Proceedings of The 12th International Workshop on Semantic*

Evaluation, pages 39–50, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[181] B. Vargas-Govea, G. González-Serna, and R. Ponce-Medellın. Effects of relevant contextual features in the performance of a restaurant recommender system. *ACM RecSys*, 11(592):56, 2011.

[182] S. Vashishth, M. Bhandari, P. Yadav, P. Rai, C. Bhattacharyya, and P. Talukdar. Incorporating syntactic and semantic information in word embeddings using graph convolutional networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3308–3318, Florence, Italy, July 2019. Association for Computational Linguistics.

[183] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[184] I. Vulić, S. Baker, E. M. Ponti, U. Petti, I. Leviant, K. Wing, O. Majewska, E. Bar, M. Malone, T. Poibeau, R. Reichart, and A. Korhonen. Multi-simlex: A large-scale evaluation of multilingual and cross-lingual lexical semantic similarity, 2020.

[185] K. Wakil, R. Bakhtyar, K. Ali, and K. Alaadin. Improving Web Movie Recom-

mender System Based on Emotions. *International Journal of Advanced Computer Science and Applications*, 6(2):9, 2015.

[186] H. G. Wallbott and K. R. Scherer. How universal and specific is emotional experience? Evidence from 27 countries on five continents. *Social Science Information*, 25(4):763–795, Dec. 1986.

[187] B. Wang and C. C. J. Kuo. Sbert-wk: A sentence embedding method by dissecting bert-based word models, 2020.

[188] T. Wharton. Interjections, language and the 'showing/saying' continuum. *Pragmatics and Cognition*, 11(1):39–91, 1 2003.

[189] C. Wu, F. Wu, S. Wu, J. Liu, Z. Yuan, and Y. Huang. THU_NGN at SemEval-2018 task 3: Tweet irony detection with densely connected LSTM and multi-task learning. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 51–56, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[190] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and

J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[191] H. Xie and T. Y. Tang. Vector Projection on Lyrics and User Comments for a Lightweight Emotion-aware Chinese Music Recommendation System. In *Proceedings of 2018 International Conference on Big Data Technologies*, ICBDT '18, pages 88–94, New York, NY, USA, 2018. ACM. event-place: Hangzhou, China.

[192] C. Xu, P. Zhao, Y. Liu, J. Xu, V. S. S.Sheng, Z. Cui, X. Zhou, and H. Xiong. Recurrent convolutional neural network for sequential recommendation. In *The World Wide Web Conference*, WWW '19, page 3398–3404, New York, NY, USA, 2019. Association for Computing Machinery.

[193] H. Xu, M. Lan, and Y. Wu. ECNU at SemEval-2018 task 1: Emotion intensity prediction using effective features and machine learning models. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 231–235, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[194] H. Xu, B. Liu, L. Shu, and P. S. Yu. Bert post-training for review reading comprehension and aspect-based sentiment analysis. In *NAACL-HLT*, 2019.

[195] H. Xu, W. Yang, and J. Wang. Hierarchical emotion classification and emotion component analysis on chinese micro-blog posts. *Expert Syst. Appl.*, 42:8745–8752, 2015.

[196] P. Xu, A. Madotto, C.-S. Wu, J. H. Park, and P. Fung. Emo2Vec: Learning generalized emotion representation by multi-task training. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 292–298, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics.

[197] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen. Deep Matrix Factorization Models for Recommender Systems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 3203–3209, Melbourne, Australia, Aug. 2017. International Joint Conferences on Artificial Intelligence Organization.

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

[199] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California, June 2016. Association for Computational Linguistics.

[200] H. Ying, F. Zhuang, F. Zhang, Y. Liu, G. Xu, X. Xie, H. Xiong, and J. Wu. Sequential recommender system based on hierarchical attention networks. In *Proceedings*

of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, pages 3926–3932. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[201] E. Zangerle, C. Chen, M. Tsai, and Y. Yang. Leveraging Affective Hashtags for Ranking Music Recommendations. *IEEE Transactions on Affective Computing*, pages 1–1, 2018.

[202] M. Zhang, Y. Zhang, and G. Fu. Tweet sarcasm detection using deep neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2449–2460, Osaka, Japan, Dec. 2016. The COLING 2016 Organizing Committee.

[203] S. Zhang, Y. Tay, L. Yao, and A. Sun. Next item recommendation with self-attention. *ArXiv*, abs/1808.06414, 2018.

[204] S. Zhang, X. Zhang, J. Chan, and P. Rosso. Irony detection via sentiment-based transfer learning. *Information Processing & Management*, 56(5):1633–1644, Sept. 2019.

[205] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. Bertscore: Evaluating text generation with bert, 2019.

[206] Z. Zhang, M. Dong, and S. S. Ge. Emotion analysis of children's stories with con-

text information. *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*, pages 1–7, 2014.

[207] Q. Zhao, Y. Shi, and L. Hong. GB-CENT: Gradient Boosted Categorical Embedding and Numerical Trees. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1311–1319, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee. event-place: Perth, Australia.

[208] W. Zhao, M. Peyrard, F. Liu, Y. Gao, C. M. Meyer, and S. Eger. Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[209] A. Zheng and A. Casari. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. "O'Reilly Media, Inc.", Mar. 2018. Google-Books-ID: sthSDwAAQBAJ.

[210] Y. Zheng, R. Burke, and B. Mobasher. The Role of Emotions in Context-aware Recommendation. In *RecSys workshop in conjunction with the 7th ACM conference on Recommender Systems*, page 8, Hong Kong, China., Oct. 2013. RecSys workshop in conjunction with the 7th ACM conference on Recommender Systems.

[211] X. Zhu, H. Guo, S. Mohammad, and S. Kiritchenko. An empirical study on the effect of negation words on sentiment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 304–313, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[212] M. Zihayat, A. Ayanso, X. Zhao, H. Davoudi, and A. An. A utility-based news recommendation system. *Decision Support Systems*, 117:14–27, 2019.