MACHINE LEARNING OVER USER-GENERATED CONTENT:
FROM UNSUPERVISED USER BEHAVIORAL MODELS TO
EMOTION RECOGNITION VIA DEEP LEARNING

by

Zahra Rajabi
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

_____	Dr. Amarda Shehu, Dissertation Director
_____	Dr. Ozlem Uzuner, Committee Member
_____	Dr. Antonios Anastasopoulos, Committee Member
_____	Dr. Liang Zhao, Committee Member
_____	Dr. Ozlem Uzuner, Department Chair
_____	Dr. Kenneth S. Ball, Dean, The College of Engineering

Date: _____ Spring Semester 2021
George Mason University
Fairfax, VA

Machine Learning over User-generated Content: From Unsupervised User Behavioral Models to Emotion Recognition via Deep Learning

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Zahra Rajabi
Master of Science
George Mason University, 2018
Bachelor of Science
Shahid Beheshti University, 2005

Director: Dr. Amarda Shehu, Professor
Department of Information Sciences and Technology

Spring Semester 2021
George Mason University

Dedication

First and foremost, I have to thank my parents for their love and support throughout my life, and for giving me strength to reach for the stars and chase my dreams. My little brother who deserve my wholehearted thanks as well to be always beside me and taking care of my parents while I was studying abroad.

Acknowledgments

I would like to sincerely thank my supervisor, Professor Shehu, for her guidance and support throughout this study, and especially for her confidence in me. Her insightful and detailed comments about my work have always been immensely helpful. I would also like to thank Prof. Uzuner for serving as co-director of my PhD dissertation Committee. I would like to thank Prof. Antonios Anastasopoulos and Prof. Liang Zhao for their willingness to be in my PhD dissertation Committee and provide me valuable feedback.

Table of Contents

List of Tables

List of Figures

Abstract

MACHINE LEARNING OVER USER-GENERATED CONTENT: FROM UNSUPERVISED USER BEHAVIORAL MODELS TO EMOTION RECOGNITION VIA DEEP LEARNING

Zahra Rajabi, PhD

George Mason University,

Dissertation Director: Dr. Amarda Shehu

We are now a machine-mediated society, with virtually every transaction and decision mediated, analyzed, and even recommended by algorithms running in the background. An increasing percentage of our interactions and societal discourse now happen over social media platforms. While this has increased our connectivity, transcending geopolitical borders, it has also provided us with unique problems that we would not have anticipated even a decade ago. The propagation of misinformation on social networks is now a societal problem and is prompting an increasing body of research into how to identify misinformation, how to identify spreaders of such information and, perhaps more importantly, how to design mitigation and intervention strategies. In this dissertation we show that advancing research to address these challenges requires foundational research into automated user behavioral profiles. We present an unsupervised learning algorithm that leverages user-generated content to understand and categorize user behavior in near-real time. This line of work rests fundamentally on the ability of machines to understand language. This, in itself, is a multi-faceted challenge that has spawned entire domains in computer science, such as natural language processing. To advance the ability of machines to understand content, we focus on another fundamental problem, emotion recognition from short text in platforms, such as

Twitter and Reddit. We significantly advance this line of research by going beyond binary sentiments and present sophisticated deep neural network-based models that can capture fine-grained emotions in the presence of a host of challenges, including data imbalance and noise due to human annotations. Finally, we ask a fundamental question of whether text is sufficient to understand emotions and show that supplementing text with new modes of interaction, such as emojis and emoticons, advances the ability of machines to disambiguate emotions, much like humans. We believe that the research presented in this dissertation lays the groundwork for further advancing machine understanding of human behavior in social media platforms.

Chapter 1: Introduction

In her book "The Social Machine," [4], Judith Donath presents a deep evaluation of what she labels as "thinking machines". Computers, the author posits, have come a long way from the thinking machines in the twenty first century and have instead become social machines. By this she refers to the online places where people meet friends, play games, and collaborate on projects. Never has this been more evident than during the COVID-19 pandemic, when virtually every social interaction, locked out from physical, shared spaces, moved to online platforms, such as Facebok, Twitter, Instagram, Redditt, and others.

In her book, Donath identifies what is needed for social media to become *sociable* media. She argues that we must design interfaces that reflect how we understand and respond to the social world. In her quest for such spaces, she articulates what is missing from current ones: people and their actions remain harder to perceive online than face to face; interfaces are clunky, and one has less sense of other people's character and intentions, where they congregate, what they do, and more.

Indeed, a growing number of social scientists are taking a deep look into how machines are changing the way we interact, and whether those changes are all that desirable. Populations are growing, and online, social media platforms have been instrumental at allowing societal discourse to happen at scale. However, some of the ills of the online discourse have already been exposed. The propagation of misinformation on social networks is now a societal problem and is prompting an increasing body of interdisciplinary, multi-faceted research into what misinformation threatens in our societies, how to identify misinformation, how to identify spreaders of such information and, perhaps more importantly, how to design effective mitigation and intervention strategies. To this day, we do not have good answers, though machine learning (ML) literature abounds. In part motivated by the premise that user activities in online platforms reveal much about who spreads misinformation and how,

ML research has focused heavily on data-driven approaches that distill user activities into useful fingerprints and models. Design of mitigation and intervention strategies has received far less attention, in part due to the challenge of designing relevant user behavior models.

In this dissertation, we lay the groundwork towards user behavior models and present a novel, data-driven approach for user behavior analysis and characterization. Namely, we leverage unsupervised learning, which relieves us from the challenge of having to obtain user characterizations (labels) by human annotators, a dubious process rife with human errors and limited in feasibility. In chapter 2, we present an unsupervised learning approach that identifies user behavioral categories over key behavior dimensions. These categories are related to content-based, user-based, and network-based features that can be extracted in near-real time. Predictive models that leverage these features are then built and evaluated in a rigorous manner in their ability to predicting user behavior from recent activity. The main contribution of this line of research is that these models can be employed to rapidly identify users for intervention in mitigation strategies, crisis communication, and brand management. We addressed the problem of quantifying user behaviors to user categories in recently published paper [5] with the aim to help mitigate fake information propagation. Understanding user behavior, while crucial, is still far off from Donath's vision of sociable machines. Indeed, the online infrastructure that supports our interactions makes it much harder to understand one another. When limited to text, which is the main media of most of our interactions online, it is hard to understand even basic emotions. The same sentence can be interpreted in many different ways. Consider the simple statement "I was so happy to see that." Does this sentence mean the individual is happy, relieved, or cynical about what must have happened prior to this statement? It is indeed hard to decode. Human language is ambiguous. Yet, an entire domain in computer science is focused on understanding human language in unambiguous terms; that is, by a machine.

Leaving aside the question of whether a machine can unambiguously decode a fundamentally ambiguous mode of communication, much of Natural Language Processing (NLP) seeks to devise artificial intelligence (AI) methodologies to disambiguate text and extract

from it functional information. Sometimes the success of such NLP models results in a hype that they truly understand or capture meaning; however, learning form is not sufficient to learn meaning and there should be a distinction that helps better natural language understanding (NLU) [6]. NLP is by now a broad domain, with diverse interests and subcommunities, many of which intersect significantly with other domains and disciplines, such as health informatics, social sciences, and more. However, more to Judith Donath's point, how can one address in some part the challenge of understanding one another online? It would be a stretch to posit that the author thinks machines can assist with that, but that is exactly what many computer scientists, NLP researchers more narrowly, are aiming to do. There is good reason for that.

We are now a machine-mediated society and well on our way to having every transaction and decision mediated by AI algorithms running in the background. In part because we have grown too big, there are now AI platforms allowing companies to track the well-being, engagement, and other behavioral aspects of their employees, as they engage with one another on social media channels, such as Slack (and others). The same AI platforms are redesigned to evaluate consumer satisfaction, track consumer interest in near-real time, distill that information for companies to make key business decisions, and even customize user experiences via personalized recommendations. Machine-mediated interactions are increasing beyond companies. For instance, they are now assisting with mental health diagnosis, providing meaningful conversation with neurodiverse persons, and even enhancing and transforming educational and learning experiences [7–12].

These technologies are moving forward at an increasing pace, and they are doing so all the while challenging NLP researchers to truly equip machines with true natural language understanding. Emotion recognition is a central aspect of this objective and very much a foundational test. Can machines understand human emotion?

The rest of the chapters in this dissertation aim to provide some answers to this key question. In this line of work, we focus on emotion recognition from short text. One reason for this is because most of our interactions happen on platforms that limit how much we

can write. Another, perhaps more profound reason is that these platforms have changed the nature of our discourse.

In chapter 3 we setup the problem of emotion recognition from short text (Twitter instances) as a supervised learning problem. In a departure from most prior ML work, which simplifies the problem into a recognition of sentiments (positive, negative, and maybe allowing for neutral), we consider fine-grained emotions, inspired in part by noted psychologists and their models. Leveraging developments in NLP regarding word representations and embeddings, preliminaries of which are provided in the Appendix, we design deep learning models that are first trained on human-labeled data and then used to make predictions. A concise version of this work has been recently published in [13] and shows that indeed, machines (that is, our deep learning models) are able to recognize emotions in text.

Chapter 3 makes an important contribution. It shows that the problem of emotion recognition from short, informal text can be formulated as a multi-label classification problem. A later chapter, Chapter 5, poses the problem additionally as a multi-class classification problem. The distinction is important, as multi-label classification allows us to recognize more than one emotion in a piece of text. This is closer to what humans do; they use their rich language to convey multiple emotional states. One can be happy and relieved at the same time.

So then, where is the challenge with machines recognizing human emotions? Where does the ambiguity of language enter the picture? The answer to this may be not quite intuitive. We hinted that the deep learning models that we contribute in Chapter 3 are trained over human-labeled data. While it is easy for us to accept that machines may have difficulty recognizing emotions from text, it may not immediately occur to us that humans may just as much have difficulty recognizing emotions. This is a key point to make. When we turn our attention to the question of what emotion(s) is/are present in the instance "I was so happy to hear that," the answers can vary. So, if this instance is part of the training dataset, it may not be labeled with the correct emotion. It does not even have to be an ambiguous sentence to throw off any one of us. Indeed, human error is very much a

recognized issue and goes beyond the specific problem of emotion recognition and pervades the use of labeled training datasets in ML.

For various reasons, human annotation error included, the training datasets that are available to us to train ML models, suffer from an issue known as data imbalance; that is, some emotions are scarce. Not all emotion classes/categories have similar number of instances/text. This presents a fundamental challenge to ML models, that goes beyond our problem of interest. Most (supervised) ML algorithms are designed around the assumption that each class in the training dataset has an equal or at least similar number of examples for each class. When the dataset does not meet this assumption, the resulting models have poor predictive performance for the minority classes; that is, the classes with very little data/instances in them. This is definitely the case with datasets labeled with emotions. Some emotions are very scarce, and an ML model does not have the opportunity to learn them well to be able to predict them reliably when given unlabeled text.

In chapter 4 we address the issue of data imbalance for emotion classification. We do so for BERT-based (deep neural network) models, which leverage the latest developments in NLP that allow better embedding of words, contextualized by neighboring words upstream and downstream in text. We identify an important knob that is common to all deep neural networks, the loss function, and investigate the impact of variations in such functions in controlling the ability of the model to better learn the underrepresented classes. The promising results prompt us to consider a broader search over not only loss function variations, but also other hyperparameters. We search over a complex hyperparameter space and identify a best model that best addresses the issue of data imbalance, and show the effectiveness of such a model and its superiority over the state of the art in various highly-granular and unbalanced datasets.

Chapter 5 provides an additional facet of the challenge of data imbalance confounded with noise and addresses it via novel techniques based on what is known as stacked generalization. The latter falls under the umbrella of meta learning, learning about learning.

The work presented in Chapter 5 provides evidence that meta-learning is a powerful framework to address data imbalance in emotion classification. So, in Chapter 6, we bring the various facets of our work so far and leverage ensemble learning over powerful NLP models to further advance the performance of emotion classification models. Broadly, ensemble learning is the process by which multiple models, classifiers in our case, are strategically built and combined to better solve a particular ML problem. We present and evaluate various strategies over shallow and deep models and contribute several that outperform our baseline models presented in the earlier chapters.

This brings us to a fundamental question in machine learning. If one thinks of deploying learned models in an actual platform to support specific tasks, how likely is it that a model trained on a curated, labeled dataset, will do a good job on new instances? This question asks about the ability of a model to generalize. We develop this question further in Chapter 7, where we pose that one way to improve model generalizibility can be via a popular approach known as transfer learning. While the current variation in emotion annotation scheme does not quite permit investigating this direction, we suspect that transfer learning will bring models closer to mature technologies that are ready for deployment.

Back to Donath's quest for sociable machines. We are still far away from online interfaces that reflect how we understand and respond to the social world. The work presented in this dissertation addresses fundamental, basic research challenges. However, moving closer to those sociable machines will require integrating mature AI platforms that are reliable, generalize well, and have the ability to keep improving. Chapter 7 articulates some important avenues for further research in this direction and posits that the work in this dissertation is an important piece of the groundwork needed to further advance machine understanding of human behavior and interactions in social media platforms.

Chapter 2: User Behavior Modelling for Fake Information Mitigation on Social Web

2.1 Summary

The propagation of fake information on social networks is now a societal problem. Design of mitigation and intervention strategies for fake information has received less attention in social media research, mainly due to the challenge of designing relevant user behavior models. In this chapter we lay the groundwork towards such models and present a novel, data-driven approach for user behavior analysis and characterization. We leverage unsupervised learning to define user behavioral categories over key behavior dimensions. We then relate these categories to content-based, user-based, and network-based features that can be extracted in near-real time and identify the most discriminative features. Finally, we build predictive models via supervised learning that leverage these features to determine a user's behavior category. Rigorous evaluation indicates that the constructed models can be valuable in predicting user behavior from recent activity. These models can be employed to rapidly identify users for intervention in mitigation strategies, crisis communication, and brand management. The results of this study have been presented in Proceedings of 12th International Conference on Social Computing, Behavioral-Cultural Modeling, Prediction and Behavior Representation in Modeling and Simulation [5].

2.2 Introduction

In 2017, 67% of Americans reported that they obtained at least some part of their news on social media.[1] This massive burst of data is naturally accompanied with the threat of disinformation, such as spam and fake news spread by malicious intent users, affecting different aspects of democracy, journalism, and freedom of expression [14]. Fake news dissemination was best highlighted during the 2016 presidential election, in which the spread of fake stories favoring each party gravely threatened trust in government [15]. The propagation of fake information on social networks is now a recognized societal problem [16]. Despite many efforts, social networking platforms have yet to effectively address this challenge [2]. In particular, mitigating the dissemination of fake content is now a critical challenge for researchers across academia leading to emerging research areas of social cyber-security [17] and social cyber forensics [18, 19].

In this chapter, we present a novel, data-driven approach for user behavioral analysis and characterization that enables us to identify vulnerable users for fake news mitigation. Our main contributions are: a.) Identification of key user behavior dimensions in reactions to the exposure of fake vs. fact information, specifically *initiation*, *propagation*, and *reception* behavior types. These dimensions allow us to organize users in behavior categories via unsupervised learning. b.) Validation of the hypothesis that behavior categories for users can be predicted by features extracted from shared content, user profile and activity, as well as the structural characteristics in the corresponding user interaction network. We also employ feature selection approaches to analyze the significance of our content-based, user-based, and network-based set of features to identify the most representative features of user behavior categories. These features are then used to build classification models to predict such categories. c.) Extensive experiments to evaluate state-of-the-art multi-class classification algorithms for user behavioral pattern prediction using a dataset collected from *Hoaxy* [20] platform. Our evaluations show that the predictive models demonstrate

[1] http://www.journalism.org/2017/09/07/news-use-across-social-media-platforms-2017/

[2] https://www.theguardian.com/technology/2018/mar/11/tim-berners-lee-tech-companies-regulations

promising performance in categorizing users based on their reactions in response to fake/fact exposures, which consequently gives us an oversight to develop a solid baseline for designing an effective fake content mitigation strategy.

2.3 Related Work

In recent years, with increasing consumption of news over social media, the extreme consequences of fake information dissemination, from misleading election campaigns to inciting violence during crises, have led many researchers to focus on the problem of fake news detection [20–22]. Comprehensive reviews of this area of research in [16,23] show that existing studies mostly rely on static datasets to develop models based on supervised learning methods rather than online learning settings due to potential concept drifts. These approaches involve exploitation of user, content, and network-based information which inspired us in the feature design of our user behavioral modeling.

Users play a critical role as the creators and spreaders of fake content in social web. Therefore, assessing the credibility of users and modeling their behavior types could provide a valuable approach to design intervention strategies [24, 25], which can optimize the dissemination of real news. For instance, [25] proposed an intervention framework using multivariate point process, however, authors did not consider the types of users and their behaviors. Given the uncertainty of user intent and activities, it is essential, although very challenging, to discriminate between malicious and naive users who unintentionally engage in fake content propagation. Therefore, modeling user behavior for identifying candidate vulnerable users for intervention strategies is an emerging research need.

While there exist extensive research on social media on user modeling and user credibility [23, 26–28], the main goal of these studies has centered around content filtering for spam, bot detection [29, 30], improving user interest profiling for content and link recommendation systems, personalization in search, as well as influencer ranking. Our research instead complements such user modeling research by investigating user behavior types to inform the mitigation strategies for the propagation of malicious, fake content.

2.4 Methods

We first represent the key behavior dimensions relevant to the mitigation task and then describe the unsupervised learning setup that allows elucidating the organization of users in different behavior categories. Once such categories are identified, information theoretic measures expose characteristics/features that best relate with the identified categories. Supervised learning methods then yield predictive models of behavior categories from such features.

2.4.1 Key Dimensions of User Behavior

We have identified three major behavioral dimensions to capture user reactions to fake over fact cascade exposures, namely initiating, propagating, and receiving (but no further action) fake content. We define $\texttt{FoF}_{\text{prop}}$, $\texttt{FoF}_{\text{received}}$, and $\texttt{FoF}_{\text{init}}$ to be log-ratio of fake over factual information respectively propagated, received or initiated by a user. As described in Section 2.5, these dimensions allow visualization of a three-dimensional semantic space as a baseline representation of user engagements within a network in response to different information cascade exposures. More importantly, they facilitate the application of unsupervised learning methods to identify behavioral categories based on user engagement in spread of fake versus fact cascades.

2.4.2 Identifying Behavior Categories via Unsupervised Learning

Clustering algorithms can group and categorize users within the three-dimensional space defined above. We consider several clustering algorithms such as kmeans, Agglomerative clustering, DBScan, and spectral clustering (*c.f.* survey on algorithms in [31]). In Section 2.5, we evaluate the performance of clustering algorithms (over different parameter values) along popular metrics, such as the Silhouette coefficient, the Calinski-Harabaz score, and the Davies-Bouldin score. These metrics do not rely on ground truth availability, which we lack.

Clustering Performance Evaluation.

The Silhouette coefficient is calculated as $(b-a)/max(a,b)$, where a is the mean intra-cluster distance, and b is the mean nearest-cluster distance for each sample (user). This metric is computed as the mean Silhouette Coefficient of all samples ranging from -1 (worst) to 1 (best). The Calinski-Harabaz score, also known as the variance ratio criterion, is defined as the ratio between the within-cluster dispersion and the between-cluster dispersion. A higher Calinski-Harabaz score indicates a model with better-defined clusters. Unlike the Silhouette score, the Calinski-Harabaz score is unbounded; the higher the score the better the cluster separation. The Davies-Bouldin is defined as the ratio of within-cluster distances to between-cluster distances and is bounded in $[0, 1]$. A lower score is better.

2.4.3 User Behavior Categories Representation

The above evaluation measures highlight the most effective clustering method and corresponding behavior categories, which can be used to label users. Our dataset has only ground truth for fake/fact content and no user labels for our analysis is provided. Therefore, our proposed approach of automatically labeling users with behavioral categories opens a way to supervised learning models that relate features to the discovered behavior categories/classes for users. Information-theoretic measures, such as Mutual Information (MI) measure is employed to evaluate characteristics/features of user nodes in the diffusion cascades that best relate with the identified behavior classes. A feature selection algorithm is employed to identify the most important features, and supervised learning methods are then utilized to build predictive models of behavior categories from the extracted features. In this section, we propose a data mining framework for user behavior category representation and prediction.

Features Extraction

The features representing each user are extracted using the propagated retweets' content, user profiles, and network structure.

1. ***Content-based Features***:

   - `sentiment`: the average sentiment intensity score of tweet texts shared by user is computed using Sentiment Analyzer tool in *nltk* (Natural Language Toolkit) Sentiment package [32].
   - `Tweet text length`: the average length of tweet texts shared by user.

2. ***User-based Features***:

   - `followers_count`: the number of users following a user; it shows a Twitter account's popularity;
   - `friends_count`: the number of users a user is following; it informs the user's interest-driven participation;
   - `influence_score`: the social reputation of each user based on follower and following counts that is computed by $\log((1+\text{followers\_count})^2 + \log(\text{statuses\_count}) - \log(\text{friends\_count}))$;
   - `listed_count`: the number of public lists of which a user is a member;
   - `statuses_count`: the number of tweets and retweets shared by a user;
   - `has_url`: a boolean feature showing if a user has a url or not;
   - `sociability`: the ratio of the number of friends_count to followers_count: $\log(1+\frac{1+\text{friends\_count}}{1+\text{followers\_count}})$;
   - `favorability`: ratio of the number of favorites to the total number of tweets; it informs higher engagement in contrast to just posting tweets. $\log(1+\frac{(1+\text{favourites\_count})}{(1+\text{statuses\_count})})$;
   - `survivability`: potential active existence on the platform over time, and it is measured as the difference between current timestamp and the timestamp at which a tweet is created;
   - `activeness`: the number of tweet statuses to the period of time since account creation; it determines the likelihood of a user to be active over a period of time

on average: $\log(1 + \frac{(1+\text{statuses\_count})}{(1+\text{survivability}))}$;

- `favourites_count`: the number of tweets a user has favored over time as a measure of user engagement level.

3. ***Network-based Features***:

   - `betweenness`: the normalized sum of the fraction of all-pairs' shortest paths that pass through a node/user. Betweenness values are normalized by $b = b\frac{(n-1)}{(n-2)}$ where n is the number of nodes in graph G.
   - `degree_centrality`: the fraction of nodes/users to which a particular user is connected.
   - `load_centrality`: the normalized fraction of all shortest paths that pass through a node.

Feature Ranking

Feature Selection is the process of identifying relevant features from a feature set that contribute most to the prediction variable and removing the irrelevant ones, in order to improve performance of predictive model. Features can be ranked according to different metrics, e.g. F-test statistic, Mutual Information (MI) measure, and p-value.

- *`F-value`*: ANOVA F-test statistic captures linear dependency of two random variables and computes F-value for each feature. This test measures the ratio of between-groups to within-groups variances; we note that the groups here are user behavior categories. When F-values are near 1, the null hypothesis is true (establishing independence).

- *`p-value`*: This allows determining whether the null hypothesis can be rejected with 95% confidence level (corresponding to p-values of 0.05). The smaller the p-value, the stronger the evidence to reject the null hypothesis.

- `MI measure`: This measure captures mutual dependency between random variables. MI quantifies the amount of information obtained about one random variable through

observing the other random variable, with zero value showing two random variables are independent, whereas higher values meaning higher dependency. In Section 2.5 we provide the F-value, p-value, and MI measure for each of the features.

F-value along with the p-value enables us in deciding whether results are significant enough to reject the null hypothesis. We investigated both univariate feature selection and recursive feature elimination (RFE) and in both methods, the top 1/3 of the features (5 of the 16 initial features) are very similar. In particular, RFE selects smaller sets of features recursively with the least important features pruned at each iteration. We employed an SVM classifier with linear kernel in the RFE estimator for this purpose.

User Behaviors Estimation:

In previous sections, we described how we group similar users based on their associated behaviors into three user behavioral clusters (classes). In this section, we focus on building a behavioral model that predicts a user behavior class by incorporating extracted features into supervised classification models. We consider ten different classifiers available via Python's scikit-learn [33] library such as Nearest Neighbors (k-NN), SVM with RBF kernel, Random Forest, a multilayer perceptron classifier (MLP), AdaBoost, XGBoost, Naive Bayes, Decision Tree, and Quadratic Discriminant Analysis (QDA). Each classifier is used with recommended default parameter settings and not optimized for performance. Specifically, the number of neighbors in k-NN is 3, for SVM with RBF kernel γ is chosen automatically, the maximum depth of decision tree is set to 5 both in the Decision Tree and the Random Forest classifier (`max_depth = 5`). In the latter, the number of estimators is set to 10 (`n_estimators = 10`), and the maximum number of features is set to 1 (`max_features = 1`). In the MLP classifier, settings include L2 penalty (regularization term) parameter $\alpha = 1$. In XGBoost classifier, the parameters are set as follows: `n_estimators = 100, learning_rate = 1.0, max_depth = 1, random_state = 0`.

Each classifier is trained on a balanced version of the training dataset to effectively compare performance while addressing the class imbalance. Two options are considered for this:

Balanced bagging versus SMOTE. We note that the balanced bagging effectively provides an ensemble method with each of the ten classifiers acting as the base classifier. While balanced bagging undersamples, SMOTE (Synthetic Minority Over-sampling Technique) oversamples [34].

The classification performance is evaluated using accuracy and F1 score. Accuracy evaluates the number of correct predictions over the total number of predictions, whereas the F1 score $= 2 \cdot (\text{precision} \cdot \text{recall})/(\text{precision} + \text{recall})$, where $precision = \frac{TP}{TP+FP}$ and recall/sensitivity $= \frac{TP}{TP+FN}$; TP, FP, and FN refer to the number of true positives, false positives, and false negatives, respectively. We note that in this multi-class setting, the F1 score is a micro-average; that is, contributions of all classes are aggregated to compute an average metric.

2.5 Experiments and Results

2.5.1 Experimental Setup

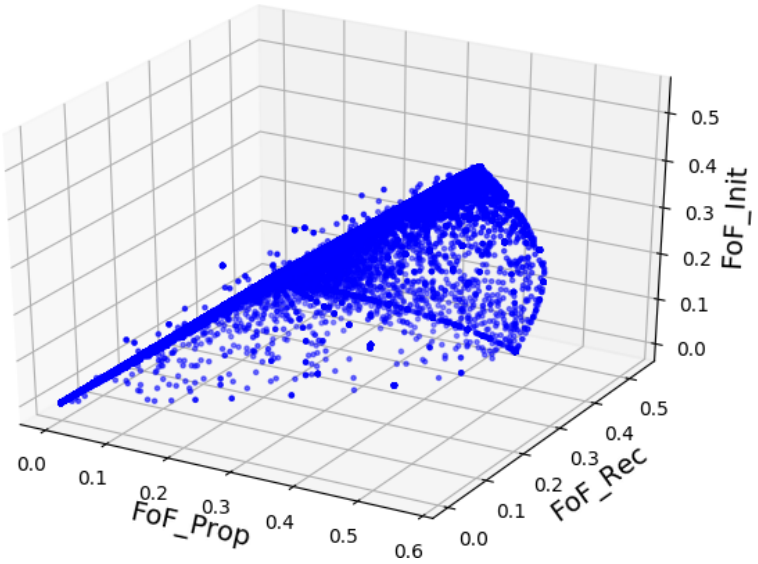
We describe three sets of experiments. First, we evaluate the performance of various clustering algorithms that allow us to learn user behavior groups (categories) in an unsupervised manner. Comparative analysis shows the most effective approach that we employ for getting the associated behavior category user labels for training. Second, we conduct a detailed selective analysis on user feature sets to choose the most relevant set for the identified behavior classes. Third, we show multi-class classification models that learn the relationship between features and the behavior classes. A principled comparison of these models along several performance metrics is presented.

Dataset: Our dataset contains records of retweets between May 16th 2016 and Dec 31st 2017 provided by [20]. Each record is a retweet of a tweet that contains at least one link to an article, which can be either a claim or a fact-checking source. The dataset consists of $20,987,210$ retweets, with $19,917,712$ (95%) linking to claim articles (fake) and $1,069,498$ (5%) to fact-checking articles (fact). We randomly sample $5,000$ users participating in

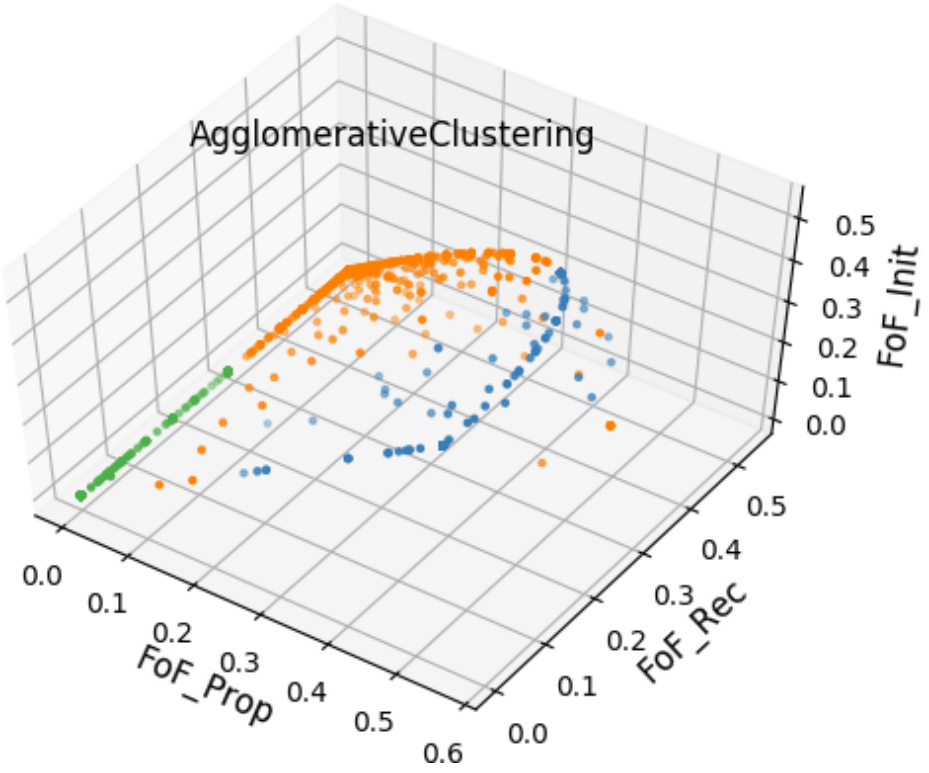
retweet cascades over which we identify behavior categories via clustering to consider as the labels (inferred ground truth) for users.

2.5.2 Visualization of User Behavior Categories

The three key behavior dimensions introduced in Section 2.4 are used to visualize the user behavior space shown in Figure 2.1(a). We observe groups of users who receive but block major fake over fact cascades, users that propagate more fakes than facts cascades, and users that act somewhere in between. These observations can be quantified via automatic groupings of users, as done via clustering algorithms. Figure 2.1(b)-(c) shows the three user behavior clusters detected by Agglomerative and kmeans clustering algorithms with the best performances and table 2.1 shows their clustering performance metrics as described in Section 2.4).

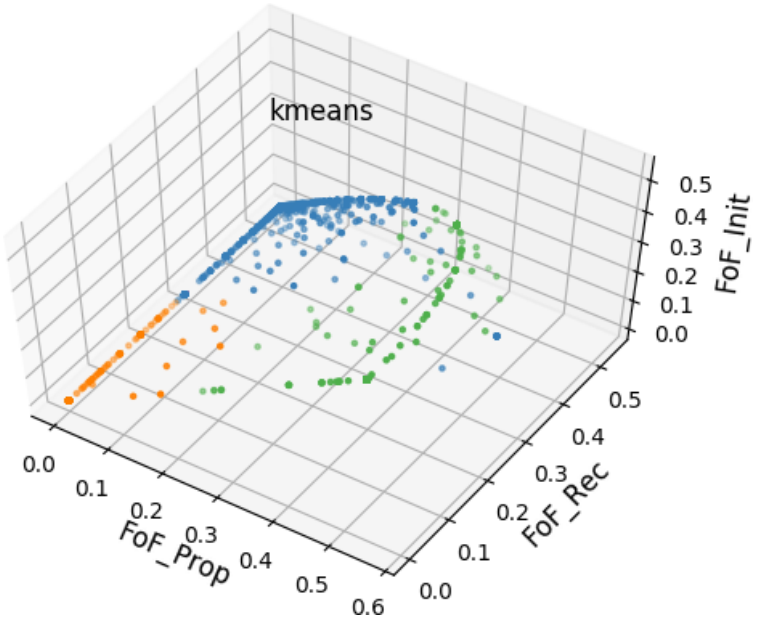


(a)



(b)

Figure 2.1: (a) Visualization of 3D user behavior space. Each dimension represents the log of ratio of user reactions (initiation, propagation, or reception) to fake over fact cascade exposures. The results of the Agglomerative clustering and kmeans clustering are shown in (b) and (c), respectively. Different colors show the emergent user clusters.



(c)

Figure 2.1: (a) Visualization of 3D user behavior space. Each dimension represents the log of ratio of user reactions (initiation, propagation, or reception) to fake over fact cascade exposures. The results of the Agglomerative clustering and kmeans clustering are shown in (b) and (c), respectively. Different colors show the emergent user clusters.

We call the three user categories detected in the semantic space of user behaviors as `malicious`, `good`, and `vulnerable/naive users`, based on their locality: `good user` has insignificant participation in the spread of fakes, `malicious user` who participate significantly in spreading or initiating fake over facts; and `vulnerable user` who mostly have lower rate of fakes to facts propagation and higher fake to fact reception. These are users who have volatile reactions in terms of behavior of reception, initiation, and propagation of fakes through the network.

Table 2.1: Comparing the performance of clustering algorithms using evaluation metrics.

Clustering algorithm	**Silhouette**	**Calinski-Harabaz**	**Davies-Bouldin**
kmeans	0.88	22913.55	0.30
Agglomerative Clustering	0.87	20951.39	0.32

2.5.3 Feature Ranking and Selection

In this section, we evaluate the significance of each feature using F-value, p-value, and MI measure calculated as described in Section 2.4 for the user behavior categories obtained via clustering. Table 5.3 shows results computed over user behavior categories obtained via Agglomerative clustering. Features with MI measure above 0.5 (important ones) are highlighted in bold. (Results computed over categories obtained via kmeans are similar and omitted due to space limit).

Table 2.2: The F-value, p-value, and MI measure for features computed over user behavior categories obtained via Agglomerative clustering.

Feature	**F-value**	**MI**	**p-value**
influence score	0.13	**0.68**	0
betweenness	0.00	0.04	0.45
deg centrality	0.07	0.06	0
clustering coefficient	0.00	0.00	0.1
load centrality	0.00	0.00	0.45
followers_count	0.01	**0.55**	0
friends_count	0.02	0.38	0
listed_count	0.00	0.39	0.02
statuses_count	0.01	**0.60**	0
has_url	0.03	0.16	0
tweet character length	0.16	**1.00**	0
sentiment	0.01	0.44	0
sociability	0.08	**0.65**	0
favorability	0.04	**0.66**	0
survivability	1.00	**0.67**	0
activeness	0.01	**0.62**	0

Feature selection by recursive feature elimination (RFE algorithm) ranks the following top five features as the most significant within each user behavior category obtained by Agglomerative clustering: *followers_count, friends_count, statuses_count, survivability* and *tweet_character_length*; And kmeans clustering: *followers_count, friends_count, listed_count, statuses_count, survivability.* We note great agreement between these two sets and the

features with MI measure higher than 0.5 shown in Table 5.3.

Visualizing Feature Profiles: Visual comparisons of feature distributions within each group can be found in Figure 2.2.

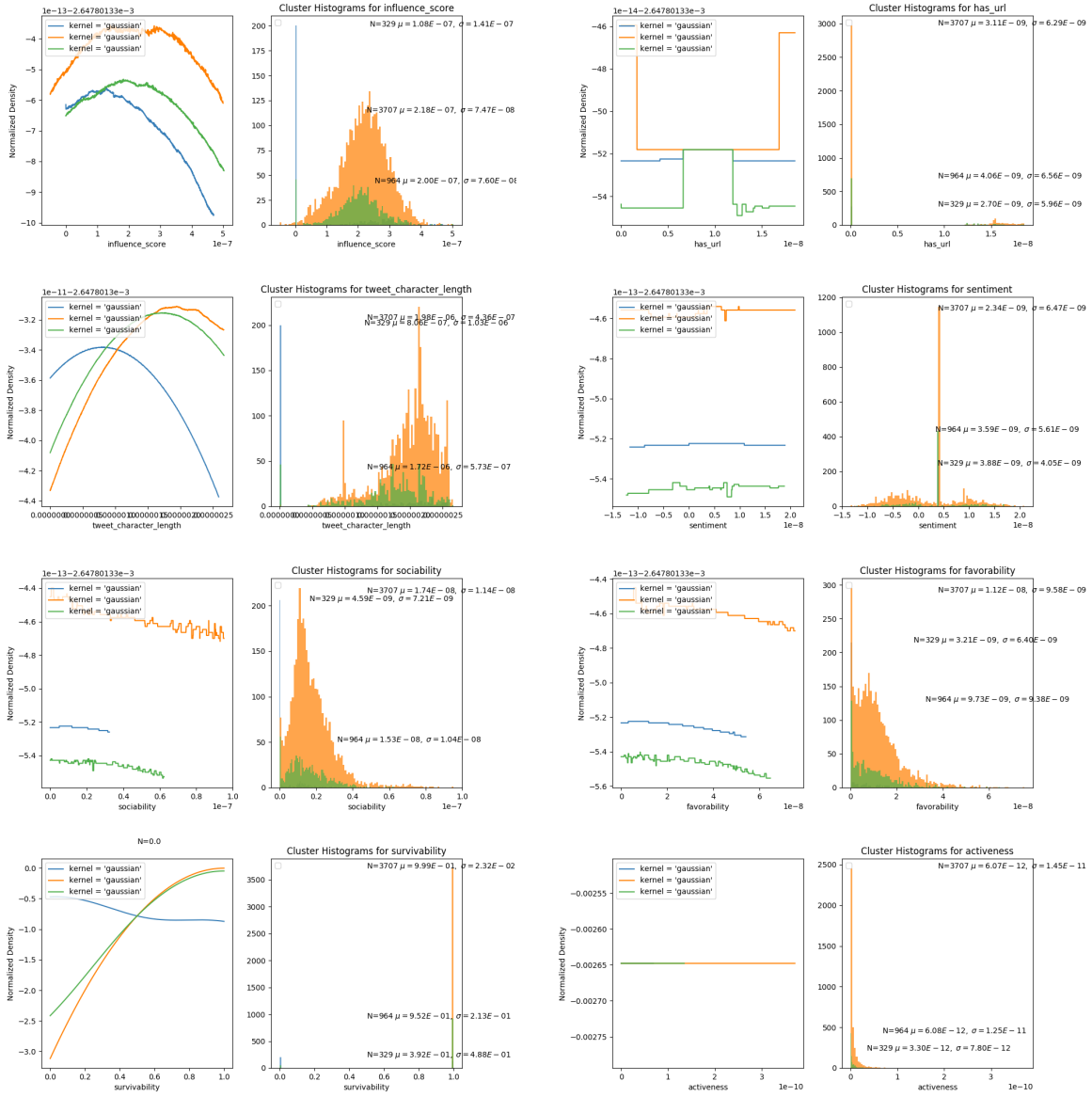


Figure 2.2: Feature distributions within each group.

2.5.4 Comparative Analysis of Predictive User Behavior Models

We examined the performance of various multi-class prediction models learned using aforementioned top features for classification of three user behavior categories obtained via both kmeans and agglomerative clusterings. A representative dataset of $5,000$ sampled users over which clustering is performed to obtain labels is subjected to both 10-fold cross validation (CV) and a $60-40$ split strategy for train-test sets. Table 2.3 shows the performance of 10 different classifiers on the test set (and average over 10 folds); the classifiers are trained over a balanced version of the training set, where we address class imbalance using balanced bagging and SMOTE sampling methods. As Table 2.3 shows, the majority of the classifiers saturate in performance around 0.80 in both accuracy and F1 score, highlighting the scope of further improvement in predicting user behavior.

Overall, the good performance belongs to MLP and SVM with balanced bagging and also k-NN classifier with balanced training set using SMOTE (`n-nearest = 3`). We also had additional experiments (results omitted due to space limit) for imbalanced settings and found both accuracy and F1 score reaching up to 0.80 for 10-fold CV. These results provide a preliminary evidence that it is possible to build user behavior predictive models that can be further improved with larger datasets or more features, by exploiting information available in near real-time for mitigation strategies.

Table 2.3: Comparison of performance of classification approaches in terms of Accuracy and F1 score for kmeans clustering labels for both 10-fold CV and split-strategy (in brackets). Notations: *BB-Acc(10-CV (Split)): Balanced Bagging Accuracy, BB-F1: Balanced Bagging F1, SMOTE-Acc: Synthetic Minority Oversampling Technique (SMOTE) Accuracy, SMOTE-F1: SMOTE F1-score.*

Classifier	**BB-Acc**	**BB-F1**	**SMOTE Acc**	**SMOTE F1**
3-NN	0.52 (0.52)	0.75 (0.51)	**0.76** (0.74)	0.76 (0.74)
RBF SVM	**0.80** (0.52)	0.81 (0.79)	0.53 (0.54)	0.53 (0.54)
Decision Tree	0.50 (0.55)	0.79 (0.52)	0.59 (0.58)	0.59 (0.58)
Random Forest	0.60 (0.54)	0.81 (0.61)	0.57 (0.56)	0.58 (0.56)
MLP	**0.80** (0.52)	0.81 (0.79)	0.53 (0.51)	0.53 (0.51)
AdaBoost	0.50 (0.36)	0.79 (0.49)	0.58 (0.57)	0.58 (0.57)
XGBoost	0.53 (0.54)	0.80 (0.54)	0.60 (0.59)	0.60 (0.59)
Gaussian NB	0.77 (0.38)	0.79 (0.76)	0.53 (0.54)	0.53 (0.54)
QDA	0.53 (0.42)	0.61 (0.62)	0.45 (0.35)	0.45 (0.35)

2.6 Conclusion and Future Work

In this chapter we have presented a novel, data-driven approach for user behavior analysis on social web for assisting fake content mitigation strategies. The identification of key behavior dimensions allows leveraging unsupervised learning to organize users along behavior categories. We identified diverse features from user information that is available in near real-time to validate predictability of user behavior categories. Supervised learning models show that user behavior categories can be predicted from such features. However, we acknowledge the limitation of the experiments, in particular, the approach to data sampling and extracted features. Given this preliminary foundation work for user modeling to serve user intervention strategies, we will address these limitations in our future work. Furthermore, behavioral psychologists can contribute detailed models of user behavior that can inform or refine the presented data-driven modeling approach. This research provides the groundwork for advanced user modeling toward mitigation-focused social cyber-security research.

Chapter 3: Beyond Binary Sentiments: A Multi-Channel BiLSTM-CNN model for Multilabel Emotion Classification of Informal Text

3.1 Summary

State-of-the-art research on emotion classification from text primarily focuses on binary or ternary classification. Yet, humans express a variety of emotions. Here we approach the classification of emotions from short, informal text as a multi-label problem, employing popular psychology models of basic and advanced human emotions. We account for imbalanced datasets differing in annotation schema, psychological models considered, and number of annotated emotions. We show that a multi-channel, multi-filter CNN-BiLSTM outperforms existing models, achieving 85.1% accuracy on the multi-label SemEval18-EC dataset. This work has been published in [13].

3.2 Introduction

While humans are inherently capable of understanding emotions encoded in text, machines have a harder time. Doing so, however, is important to achieve true natural language understanding [8, 9, 12] in order to extract meaningful user emotions in designing relevant user behavior models [35] on social web.

In this chapter, instead of simplifying emotions into two or three categories, we consider a variety of emotions based on popular psychological models. We approach emotion classification as a multi-label classification problem and focus here on the SemEval2018 Task 1 (*Affect in Tweets*) dataset. We improve emotion classification accuracy over related work by an appreciable amount. We present sub-models and various deep neural network

(NN) architectures, including a multi-channel, multi-filter BiLSTM model that outperforms existing models, achieving 85.1% accuracy on the SemEval-EC dataset.

3.3 Related Work

Work in [36] is one of the first to conduct multi-class emotion classification from text posted by social media users on Twitter. The authors consider various feature sets to classify tweets into one of 7 emotion classes: love, happiness, fun, neutral, hate, sadness, and anger. Depending on the dataset considered, the accuracy achieved in this setting varies from 56.9% to 60.2%. In contrast, on binary and ternary emotion classification, the authors report an accuracy range of 70.1% to 81.3%, highlighting intrinsic challenges with multi-class emotion classification.

Since the work in [36], great strides have been made on ML research on social media text. In particular, the Semantic Evaluation (SemEval) competition has spurred research on computational semantic analysis systems. In particular, semantic analysis was included as a semantic annotation task in SemEval 2007 (SemEval-1). The most recent evaluation of this task was SemEval-2018 Task 1 "Affect In Tweets" [37], in which five sub-categories were designated, emotion intensity regression, emotion intensity ordinal classification, valence regression, valence ordinal classification, and emotion classification. Data was made freely available, and 75 teamsparticipated. In Section 3.4, we further describe this dataset, as we employ it as well to evaluate the performance of our top models. In Section 3.5, we compare these models against the top teams that participated in SemEval-2018 Task 1 on the emotion classification sub-task.

As related in Section 3.2, we aim to detect a rich set of fine-grained emotions. Though emotion models are still debated in psychology, popular models agree on five common emotions, such as *joy*, *sadness*, *disgust*, *anger*[38–40]. One of the challenges with learning to detect emotions in text is the fact that available training data differ in psychological models employed (i.e., annotation schemes) and, more practically, the number of emotions

annotated [41].

3.4 Methods

3.4.1 Data Preprocessing

Given a dataset of sentences, we use the Keras tokenizer API to preprocess them by different methods as follows: (1) Tokenizing; (2)Stemming; (3) Removing stop-words; (4) Converting all letters to lowercase; (5) Stripping punctuation; (6) Removing accent marks and other diacritics; (7) Removing superfluous white spaces; (8) Expanding abbreviations; (9) Removing sparse terms and repeated letters; (10) Removing html links and emojis; (11) Removing mentions and usernames (@twitter_id); (12) Removing non-words and multiple letter repeating words; and (13) Removing short/long words.

The API allows us to convert a pre-processed, tokenized sentence into a sentence matrix, where the rows are word vector representations of each token. These can be outputs from the pre-trained Word2Vec [42] or GloVe [43] models. We make use of the Keras tokenization utility class to vectorize a text corpus by converting each token into a sequence of integers, where each integer is the index of a token in a dictionary. Each sequence of integers is padded and converted into fixed-length sequences using the tokenization API. The categorical labels are also encoded using sklearn's LabelEncoder and then transformed into one-hot vectors as numerical data.

3.4.2 Deep Learning Models

We construct several deep NN architectures, utilizing the Keras library with tensorflow backend [44]. We experiment with Convolutional NNs (CNNs) in combination with LSTMs. Specifically, we investigate the following five architectures.

Basic CNN

This uses one 1-D convolution layer and an l2 regularizer ($l2 = 0.001$). We utilize a basic CNN model as a baseline, motivated by its great performance on locally-structured data (such as images), the presence of local structure (dependencies between neighboring tokens), and the possible improvement in performance that encoding such structure may confer to extracting emotions from tweet sentences.

Multi-filter CNN

This allows better leveraging the contextual information by providing several convolutional filters of different stride length in $\{3, 5\}$.

Static, Multi-filter CNN with GloVe embeddings

This uses a pre-trained 100-dimensional embeddings of words as input. During training, the unknown word vectors are initialized randomly and kept static; that is, weights are not being updated. However, the rest of parameters of the model are learned.

Non-static, Multi-filter CNN with GloVe embeddings

This allows additional training and is fine-tuned via backpropagation. The model is inspired by work in [45] and is a 2-channel model with two sets of word vectors, static and non-static. Each filter is applied to both channels, but gradients are back-propagated only through one of the channels. As a result, the model can fine-tune one set of vectors while keeping the others static. Both channels are initialized by GloVe embeddings.

Multi-channel, Multi-filter CNN

This is a CNN with hyperparameter tuning and static word vectors [2]. Work in [2] presents several models: CNN-rand, a baseline model with randomly-initialized word vectors; a static CNN that outperforms the CNN-rand model; a non-static CNN model with pre-trained word vectors but fine tuned vectors; finally, Which is variant of CNN with 2 sets of word

vectors, static and non-static, to which we refer as Kim-CNN in [2]. This model outperform the CNN-rand model and non-static CNN model and is the reason we evaluate it on the Crowdflower and SemEval-EC datasets.

Multi-channel, Multi-filter CNN-BiLSTM

This model inspired by the model published in [2]. The LSTMs learn the sequential aspects of the data, whereas the CNNs extract fine-grained features. It is expected that a combination of these units will allow the model to self-learn and make overall better predictions. Two channels are employed. In the first channel, which is a static word embedding layer from pre-trained GloVe, the weights are frozen. In the second channel, back-propagation changes the weights for better generalization. As shown in Fig. 3.1, this model consists of two parallel components. In the first component, both the static and non-static embedding layers are fed into two BiLSTM layers as input which encode embedding word vectors. The output of BiLSTM units are fed into multiple convolutional layers with different filter sizes followed by dropouts which their outputs are merged. On the second component, each of the embedding layers are first fed into convolutional layers, their output are merged and then fed to BiLSTM units to gain sequential aspect of text. The output of the two components are then merged and we apply max pooling over the complete output to consolidate to a smaller dimension. The intuition behind this model is that the convolution layer will extract local features, and the LSTM layer will then be able to use the ordering of these features to learn about the input's text ordering.

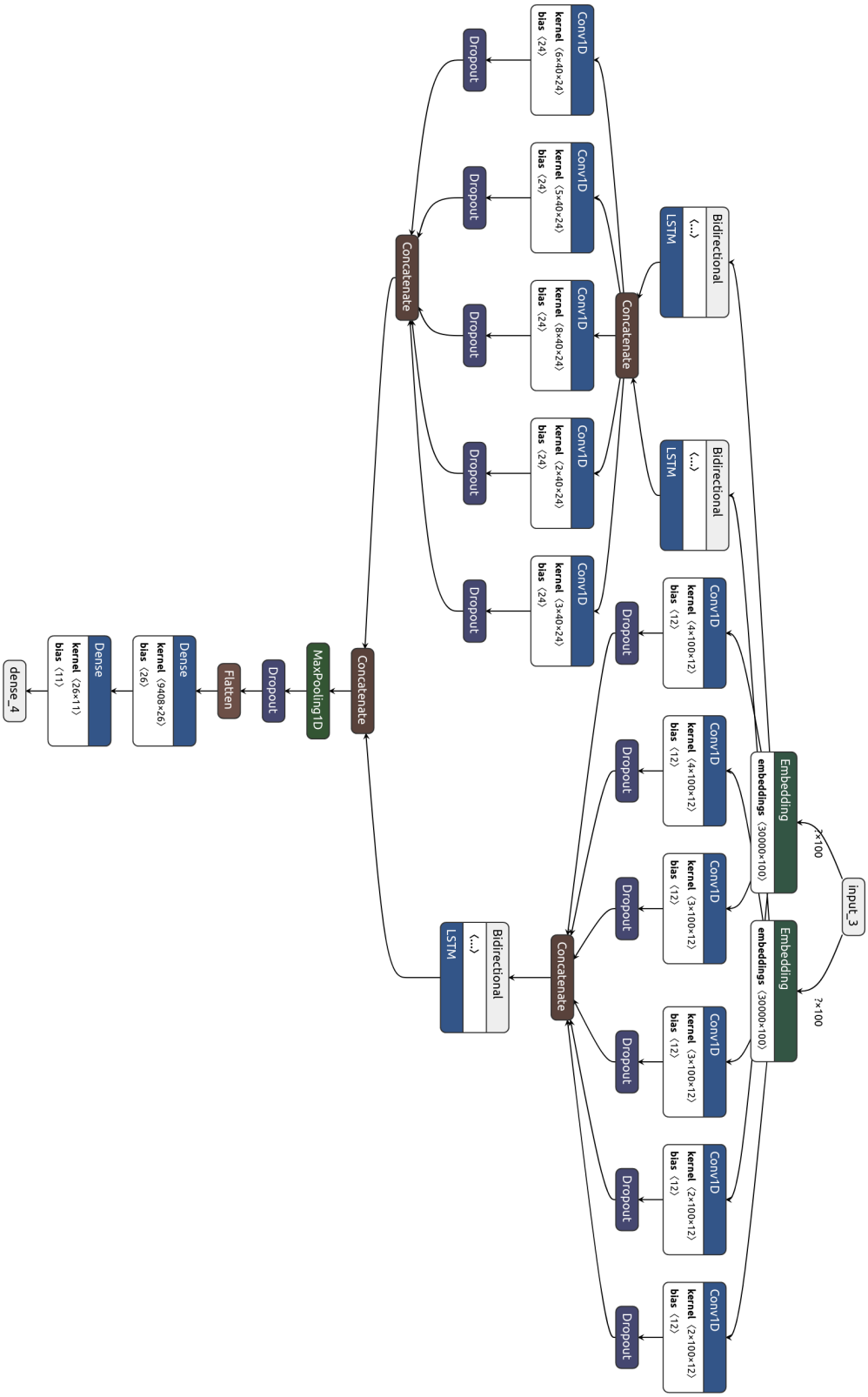


Figure 3.1: The multi-channel, multi-filter CNN-BiLSTM model inspired from work in [2].

3.4.3 Parameters of Deep NN Architectures

In all the deep NN models we investigate, the channels are concatenated and maxpoling is performed. The last layer is a fully-connected layer which produces a prediction via a sigmoid activation function. We use the rectified linear unit for the hidden layers and a binary cross_entropy loss function.

Parameters for learning are as follows:

- NB_WORDS = 30000 (Parameter indicating the number of words in the dictionary)
- VAL_SIZE = 1000 (Size of validation set)
- NB_EPOCHS = 100 (Number of epochs)
- BATCH_SIZE = 512 (Size of the batches used in the mini-batch gradient descent)
- MAX_LEN = 100 (Maximum number of words in a sequence)
- GLOVE_DIM = 100 (Number of dimensions of the GloVe word embeddings)
- MAX_SENT_LEN = 300 (character-based length)
- MAX_DOC_LEN=5 (Number of sentences in text)

3.5 Results

In the SemEval-EC dataset, each instance is annotated by multiple emotions with label 1 meaning emotion can be inferred and 0 meaning it cannot be inferred [37]. The 11 emotions in this dataset is based on Ekman's psychological model: the 8 basic emotions of *joy*, *sadness*, *fear*, *anger*, *anticipation*, *surprise*, *disgust*, and 3 additional, complex emotions of *love*, *optimism*, and *pessimism*. The dataset is split into training ($6,838$ instances/tweets), validation (886 tweets), and testing ($3,259$ tweets). The class distribution for the training set is shown in Figure 3.2. As Figure 3.2 shows, the dataset is imbalanced.

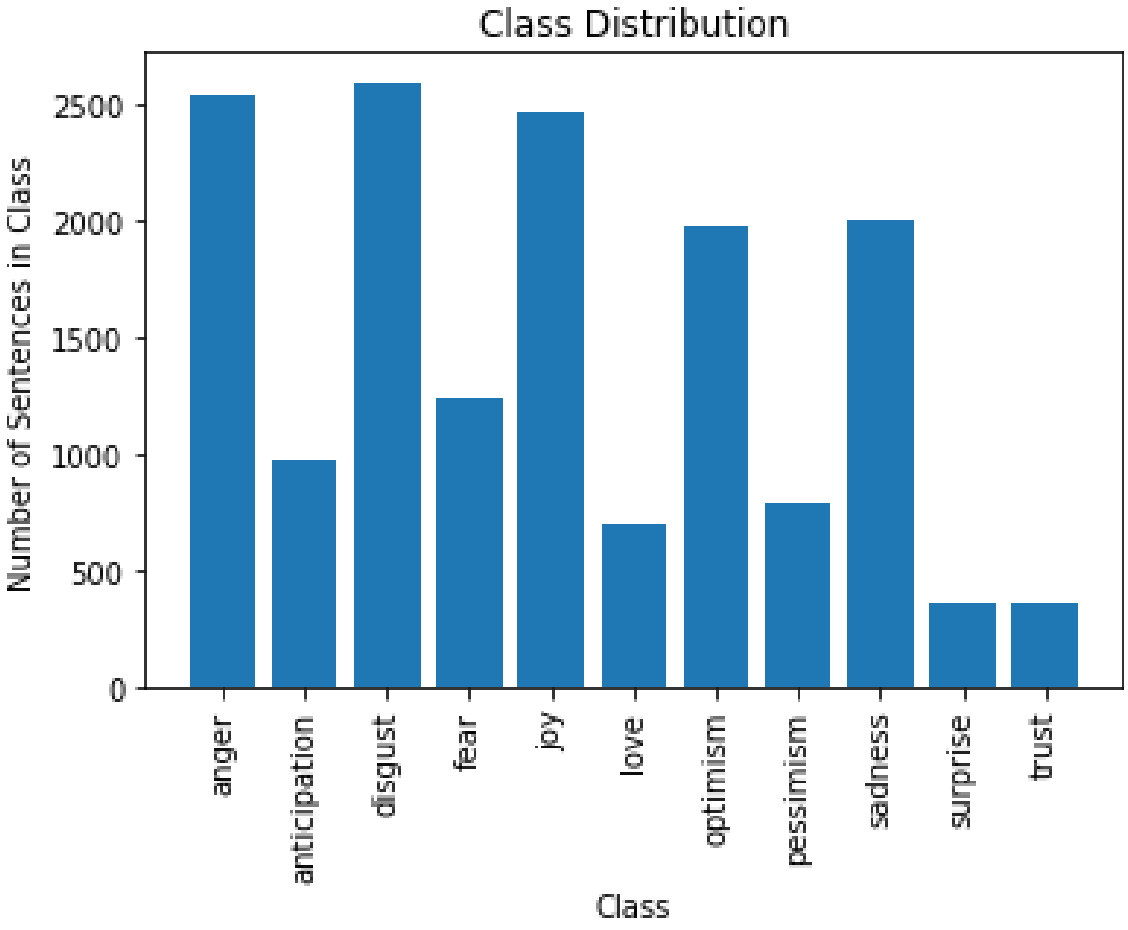


Figure 3.2: Class distribution in the SemEval-EC dataset.

3.5.1 Comparison of deep NN models on SemEval-EC dataset

The performance of the models on the SemEval-EC dataset is related in Table 3.1. Table 3.1 shows first that the models are less prone to overfitting on the SemEval dataset. The multi-filter CNN model, kim-CNN and the static multi-filter CNN model with GloVe word embeddings overfit more than the other models; their accuracy on the training dataset is about 6% higher than on the testing dataset. In contrast, the Non-Static, Multi-filter CNN achieve a similar accuracy around 81% on the training and testing datasets. In particular, the multi-channel multi-filter CNN-BiLSTM achieves the highest accuracy of about 85.1% on testing dataset and does not show overfitting problem.

Table 3.1: Comparison of performance of deep NN models on the SemEval18-EC dataset. The highest accuracy on the testing dataset is highlighted in bold font. The embedding layers were built with GloVe Embeddings.

CNN model	Train accuracy	Test accuracy	macro-F1	weighted-F1	micro-F1
Multi-filter CNN	0.876	0.814	0.43	0.53	0.54
Static, Multi-filter CNN	0.861	0.817	0.43	0.53	0.54
Non-Static, Multi-filter CNN	0.810	0.815	0.26	0.35	0.36
Multi-channel, Multi-filter CNN-BiLSTM	0.814	**0.851**	0.40	0.56	**0.60**
Multi-channel, Multi-filter CNN (Kim-CNN)	0.877	0.828	0.44	0.57	0.58

3.5.2 Comparison with the State of the Art

We now contextualize the performance of the constructed deep NN models in comparison with the top ten performers in the CodaLab competition. Table 3.2 shows the accuracy, F1-score (macro average) and F1-score (micro average) for the top ten teams. While it is not possible to obtain details into the models constructed by these teams, Table 3.2 gives insight into the state of the art. The best team achieves only an accuracy of 57.4%. In contrast, all our deep NN models shown in Table 3.1 achieve an accuracy of 81% or higher. The multi-channel multi-filter CNN-BiLSTM model accuracy is the highest over all the other models, and its F1-score (micro-average) is comparative to the best-performing models.

Table 3.2: Top Ten performers in CodaLab Competition

Team	Accuracy	F1-score (macro avg)	F1-score (micro avg)
(1)	0.574	0.574	0.697
(2)	0.574	0.500	0.689
(3)	0.566	0.490	0.673
(4)	0.562	0.549	0.678
(5)	0.558	0.488	0.674
(6)	0.558	0.476	0.677
(7)	0.552	0.512	0.658
(8)	0.539	0.543	0.664
(9)	0.516	0.435	0.640
(10)	0.501	0.467	0.631

3.5.3 Edge Case Analysis

While the above comparative results are encouraging, it is informative to understand where our models mislabels. First, we recall that the vector of possible emotions on the SemEval-EC dataset is: ['anger', 'anticipation', 'disgust', 'fear', 'joy', 'love', 'optimism', 'pessimism', 'sadness', 'surprise', 'trust'] Below we show a few mislabeled tweets: Tweet 1: "looked like weed version cousin adams family." This tweet is annotated with the emotions of *anticipation*, *joy*, and *surprise*. Our best model, the multi-channel, multi-filter CNN BiLSTM predicts the following vector of probabilities (rounded to the third decimal): [0.458 0.176 0.493 0.313 0.186 0.04 0.171 0.229 0.438 0.1012 0.047]]. No particular emotion is predicted with probability < 0.5 in this case. The two emotions with the highest probabilities are anger and disgust. What this mislabeling indicates is that this tweet is indeed a challenging case. It is reasonable to assume that even humans have a hard time correctly annotating such short text.

Tweet2: "yuko best known cheery personality dimply smile prominent squirrel teeth hates balloons oshimayuko." This tweet is annotated with the emotion *joy*. The probabilities predicted by the model are: [0.067 0.212 0.077 0.111 0.823 0.371 0.648 0.083 0.137 0.0897 0.179]. Emotions with probabilities higher than 0.5 are *joy* and *optimism*. In this case, the model correctly captures one of the emotions. This mislabeling actually highlights the ambiguity and possible overlap among the annotated emotions.

3.6 Conclusion

We have investigated several approaches to address emotion classification in short, informal (Twitter) text. We have approached emotion classification as a multi-label problem, employing popular psychology models of basic and advanced human emotions. We have presented several deep neural network architectures. Our results are encouraging; our multi-channel, multi-filter CNN-BiLSTM outperforms existing models, achieving 85.1% accuracy on the multi-label SemEval-EC dataset. Several directions emerge for future research. These

include sarcasm detection, which is recognized to be particularly challenging, multi-label classification based on prediction probabilities, as well as broadening the scope to include other datasets and online communities.

Chapter 4: Detecting Scarce Emotions Via BERT and Hyperparameter Optimization

4.1 Summary

We have formulated emotion classification as a multi-label classification problem. Under this formulation, one is confronted with the issue of data imbalance; some emotions are scarce. This chapter makes two contributions in this regard. First, it demonstrates that the BERT which makes use of transformers is now considered state of the art for emotion classification, is challenged by data imbalance. Second, it shows that data imbalance can be remedied via specialized loss functions. We investigate two main classes of loss functions, binary cross entropy and focal loss and within each evaluate the effect of various modifications to address data imbalance. Hyperparameter optimization in a resulting complex hyperparameter space reveals a best model. Our experiments on two benchmark multi-annotated datasets, GoEmotions and SemEval-EC, show that specialized loss functions significantly improve the performance of transformer models in the presence of highly imbalanced data, further advancing the state of the art in multi-label emotion classification and opening venues for further research.

4.2 Introduction

Emotion recognition from text is now a fundamental AI task across various online platforms [46]. The increasing volume of short text (tweets) in social media is allowing machine learning (ML) researchers to train increasingly sophisticated ML models for emotion recognition from text. While benchmark data now exist, ML research varies in whether emotion recognition is formulated as classification of short text into sentiments (positive, negative,

neutral), actual emotions (varying from few emotions of interest to finer-grained categories), or a regression task, where the goal is to additionally measure sentiment strength or polarity [46]. Across all these formulations, the Bidirectional Encoder Representations from Transformers (BERT) architecture has been shown to yield state-of-the-art models that outperform variations of CNN- and LSTM-based architectures [47].

This chapter makes several contributions in emotion classification (EC). We first show that the BERT architecture touted as the most significant advancement in emotion recognition from text [47,48] is challenged by data imbalance. This becomes increasingly evident when the focus is on capturing fine-grained emotions, as we show here via two benchmark datasets, GoEmotions and SemEval-EC. Both datasets broaden the formulation of emotion classification as a multi-label classification problem. The chapter shows that BERT-based models can improve performance even on imbalanced datasets when equipped with specialized loss functions. We investigate here two main, popular classes of loss functions, binary cross entropy and focal loss, and evaluate several modifications to address data imbalance, such as positive weights, class-balanced terms, and combinations.

The various decisions are treated as hyperparameters, and the resulting hyperparameter space is explored to reveal a best BERT-based model for data imbalance in EC. Detailed evaluation on the highly-imbalanced GoEmotions[48] and SemEval-EC [37] datasets relate optimal BERT-based models that better address data imbalance and so advanc multi-label EC.

4.3 Related Work

ML literature on EC has traditionally focused on sentiments and addressed binary or ternary classification problems, depending on whether neutral sentiment is considered along with negative and positive sentiments [49]. Notable psychologists have contributed to our understanding that humans are capable of expressing many emotions. Plutchik proposed eight basic emotions, *joy*, *sadness*, *fear*, *anger*, *anticipation*, *surprise*, *disgust*, and *trust* [39].

Ekman, Friesen, and Ellsworth discount anticipation and trust from the basic emotions, using universal facial expressions as the basis for this determination [38]. This disagreement has spilled over to benchmark data; different datasets have different granularities and are annotated with different sets of emotions [41].

There is great diversity in formulations of emotion recognition from text [46]. However, agreement is emerging in what is currently the state of the art (SOTA) [47]. Pre-trained BERT models [50] as a variation of transformer models have shown significant improvement over other models [47]. Unlike traditional word embeddings, such as Word2vec and Glove, which are context-dependent and have been used in Convolutional Neural Net (CNN)-based models for EC (in Kim-CNN [2] and R-CNN-BiLSTM [13]), contextual language models, such as ELMo [51] and BERT, generate word representations by taking into account both positional and contextual information. Recent work [50, 52], though limited to sentiment classification, shows that approaches based on ELMo or the Generative Pre-trained Transformer (GPT) [53] do not perform as well as BERT-based approaches. This finding has motivated many of the recent approaches to build over BERT.

However, data imbalance remains a challenge. This is not an issue unique to EC, but it becomes more acute for fine-grained human emotions. Data imbalance manifests itself in many benchmark datasets, and the two we employ here, the GoEmotions and the SemEval-EC datasets, are representative of this issue. This is not surprising; even human annotators are not as reliable in recognizing certain emotions at such granularity (as we show in Section 4.4).

Here we investigate specialized loss functions designed to handle data imbalance rather than rely on data sampling strategies. Strategies, such as oversampling using SMOTE, can increase accuracy on the smaller classes but lower the overall performance. Instead, recent ML work in [54] proposes a class-wise re-weighting scheme for most frequently used loss functions (softmax-cross-entropy, focal loss, etc.) to improve performance on data that is highly class imbalanced. We leverage these developments in this chapter.

Training a model requires tuning different hyperparameters, such as learning rate and

classification threshold; while the learning rate affects the ability of backpropagation to converge fast to a local minimum of the loss function, the classification threshold allows converting class probabilities to labels. As we describe in Section 4.4, specialized loss functions leverage different hyperparameters; by formulating hyperparameter tuning as an optimization problem, we find values that lead to an optimal model in the presence of data imbalance. In this chapter, optimality is measured with regards to macro-averaged F1 (macro-F1 for short), and optimization algorithms search for maxima of macro-F1.

Finally, we note that there are different metrics to evaluate an ML model. They are generally built over the two key concepts of precision and recall. F1-score is the harmonic average of precision and recall. Macro-F1 score is the average over the class-specific F1 scores. Weighted-F1 score weighs the contribution of each class based on the proportion of samples in it. However, since weighted-F1 score heavily biases the contribution based on the size of a class, macro-F1 score, which weights each class-specific F1-score equally, is a less-biased metric for imbalanced data. Accuracy is also an appealing metric, though not as sensitive to data imbalance as macro-F1, which is the reason we focus on optimizing macro-F1 in the search for a best BERT-based model for multi-label EC in the presence of data imbalance. We now proceed to relate methodological details.

4.4 Methodology

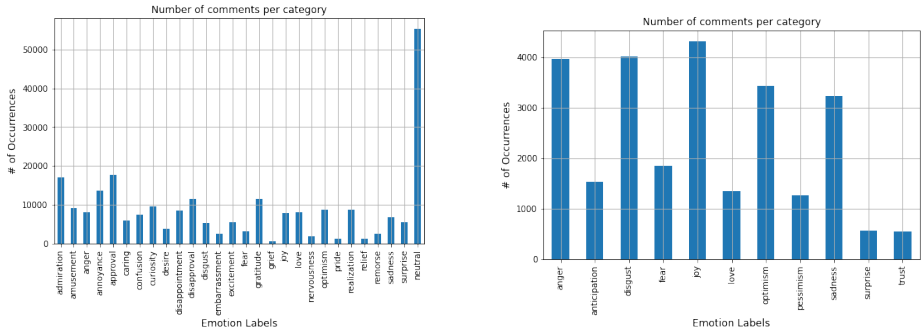
We first provide details on the GoEmotions and the SemEval-EC datasets, where we expose the data imbalance issue. The classification setup and the BERT "baseline" model are described next, followed by the specialized loss functions and the hyperparameter optimization formulation that leads to a best model in this chapter.

4.4.1 Datasets

GoEmotions Dataset The GoEmotions dataset recently released by Google research [48] contains 58K Reddit comments, each annotated with multiple emotions from a set of 27: *admiration*, *amusement*, *anger*, *annoyance*, *approval*, *caring*, *confusion*, *curiosity*, *desire*,

disappointment, *disapproval*, *disgust*, *embarrassment*, *excitement*, *fear*, *gratitude*, *grief*, *joy*, *love*, *nervousness*, *optimism*, *pride*, *realization*, *relief*, *remorse*, *sadness*, and *surprise*. It is also possible for a sample to be marked as 'neutral' or 'no emotion' if it carries none of the other 27 emotions. The dataset is split into three subsets: a training set of $43,410$ comments, a testing set of $5,427$ comments, and a validation set of $5,426$ comments. The class distribution for the training set is shown in Figure 4.1(a).

SemEval-EC Dataset In the SemEval-EC dataset [37], each of $10,983$ instances (tweets) is annotated with multiple emotions from a set of 11 based on Ekman's psychological model: the 8 basic emotions of *joy*, *sadness*, *fear*, *anger*, *anticipation*, *surprise*, *disgust*, and 3 additional, complex emotions of *love*, *optimism*, and *pessimism*. It is also possible for an instance to be annotated as 'neutral' or 'no emotion'. The dataset is split into three: a training set of $6,838$ tweets, a testing set of $3,259$ tweets, and a validation set of 886 tweets. The class distribution for the training set is shown in Figure 4.1(b).



(a) GoEmotions (b) SemEval-EC

Figure 4.1: Class distributions of training datasets.

4.4.2 Fine-tuning BERT Model

BERT for EC: In this chapter, we investigate the efficacy of *BERT-Base-Uncased pre-trained model* proposed in [50] for EC. Unlike the sequential RNN and LSTM architectures, BERT was pre-trained on a large unlabeled corpus of texts by jointly conditioning on both left and right directions. Fine-tuning involves plugging in the tasks-specific input and output

texts into pre-trained BERT and adjusting all the parameters end-to-end for 2 to 5 epochs on a supervised dataset. In this chapter, we fine-tune different BERT models using a variety of loss functions. Our baseline is the fine-tuned BERT model recently presented in [48] (only on GoEmotions dataset), to which we refer as *BERT-tuned*. For fine-tuning BERT, a dense output layer is added on top of the pre-trained model of [50] with sigmoid cross entropy loss. To support multi-label classification and independence of classes, we instead employ sigmoid binary cross entropy loss as our baseline to compute class probabilities at the final layer. However, in order to uncover underrepresented emotions, we further investigate the impact of loss functions and additional hyperparameters as mechanisms to better address class imbalance.

Loss Functions: We restrict our attention to two popular loss functions, Binary Cross Entropy (BCE) and Focal loss (FL). We recall that BCE loss for instance $n \in [N]$ is defined as $\text{BCE}(y, n) = -y_n \cdot \log \hat{y}_n + (1-y_n) \cdot \log(1-\hat{y}_n)$, where n refers to a particular instance over N instances in the dataset, y_n $in\{0, 1\}$ refers to the label, and $\hat{y_n} \in [0, 1]$ is the prediction. We note that sigmoid is the only activation function compatible with BCE loss. FL loss is also a cross-entropy loss but weighs the contribution of each sample to the loss [55]; if a sample is already classified correctly, its contribution to the loss decreases. Specifically, FL for instance n is defined as $-(1-p_n)^\gamma \cdot \log p_n$, where p_n is the predicted probability for instance n, and γ controls the attention of the model towards rare classes; $\gamma > 1$ reduces the loss for well-classified instances, where the model predicts probability > 0.5 (conversely, increases loss for hard-to-classify instances, where the model predicts probability < 0.5). When $\gamma = 0$, FL becomes equivalent to cross entropy loss.

4.4.3 Strategies to Address Data Imbalance

As summarized in Section 4.3, conventional approaches to address class imbalance use class re-balancing strategies, such as data sampling or cost-sensitive learning, where samples are weighted by inverse class frequency. Oversampling methods, may increase accuracy on smaller classes; however, as the number of samples increases, it becomes highly likely that a

newly-added sample is a near-duplicate or overlaps with existing samples. Thus, oversampling strategies often result in over-fitting. Therefore, in order to reduce the impact of class imbalance, we pursue two main strategies, true positive weighting and class balancing.

True Positive Weighting (TPW): One approach to address class imbalance is to add weights to positive samples for each label individually. TPW is different from class weights and is not calculated based on minority/ majority class, but the number of 1s vs 0s within each class. So, instead of dividing each class count by the total number of samples, as opposed to class weights, we obtain the number of 0s divided by the number of 1s for each class to give an inverse impact to positive samples within each class. BCE loss with TPW modification is defined as $\text{BCE} - \text{TPW}(y) = -w_{n,c}[p_c y_n log\sigma(y_n) + (1-y_n) \cdot log(1-\sigma(y_i)))]$, where y_n is the label, σ is the predicted probability, c is the class number, and p_c is the positive weight for the positive sample of class c.

Class Balancing: We investigate the impact of adding a class-balance term to a loss function. As introduced in [54], this term is inversely proportional to the *effective number/expected volume*, E_n, of samples. Specifically, $E_n = \frac{1-\beta^{n_i}}{1-\beta}$, where $n_i \in \mathbb{Z}$ is the number of samples in class i and $\beta \in [0, 1)$ is a hyperparameter. This formulation assumes that a new sample interacts with the volume of previously-sampled data in one of two ways, either wholly covered or wholly outside. From now on, we will refer to the class-balance term as *CB*. So, for instance, FL-CB refers to focal loss with the CB modification. Given any loss function L (BCE or FL in our case), class-balance loss can then be written as $\text{L} - \text{CB} = \frac{1-\beta}{1-\beta^{n_i}} \cdot L$. It is worth noting that work in [55] introduces an α-based variant of FL-CB, as in $-\alpha_n \cdot (1-p_n)^\gamma \cdot \log p_n$, where α_n is a parameter. This is equivalent to the above when $\alpha_n = \frac{1-\beta}{1-\beta^{n_i}}$.

4.4.4 Hyperparameter Optimization

Hyperparameter Space: Unlike work in [54], where the search space is defined over only a few parameters (namely, β and γ), we consider a more general configuration and search

over a larger and higher-dimensional hyperparameter space. Specifically, we consider the loss function itself as a categorical hyperparameter (BCE versus FL) and additionally consider four settings: 1-no strategies to address imbalance (the baseline setting), 2- adding TWP only, 3- adding CB only, and 4- adding both TPW and CB, which results in 8 different settings. Under each setting, we posit five hyperparameters. The first two are general; the threshold (THR), which allows converting class prediction probabilities to class labels), and the learning rate (LR) employed during backpropagation. The other three terms are γ, α, and β. In contrast to related literature, which limits γ and β to discrete values, we treat all real-valued hyperparameters as continuous variables that take values within a predefined range (detailed in Section 4.5.

Optimization Objective and Process: We select macro-F1 as the optimization objective, as it is more appropriate for imbalanced data. As our hyperparameters are categorical (the loss functions) and continuous (the real-valued hyperparameters), we define many parallel, independent optimization processes and then combine and compare the results to reveal a best model. We carry out a random search of the hyperparameter space. Specifically, N configurations are sampled from the space, and a BERT-based model is trained over the training dataset with a specific sampled configuration of hyperparameters. This process is carried out with the Tune library in Ray.io API [56]. The library accelerates hyperparameter optimization by providing state-of-the-art search algorithms and sampling methods. It utilizes trial schedulers to terminate bad trials (as a way of expediting search in a vast and high-dimensional space). The library also allows us to carry out several trials in parallel. In the interest of time, we limit the number of trials to $N = 50$ for the SemEval-EC dataset and to $N = 20$ for the larger GoEmotions dataset. The search is carried out over one GPU; therefore, to further lower the computational cost, we implement an *early stopping strategy*. This strategy starts with random search over the hyperparameter space but prevents the model from diverging/straggling in a large continuous search space by periodically pruning low-performing trials. In addition, we make use of the *Asynchronous Hyper Band Scheduler* which enables aggressive early stopping of bad trials and provides

better parallelism.

4.5 Experimental Results

As related in Section 4.4, we fine-tune the pretrained, *BERT-base* model presented first in [50] for our experiments. This BERT-base model is fine-tuned as in [48]. First, a dense output layer is added on top of the pre-trained model for the purpose of fine tuning, with a sigmoid for multi-label classification. The AdamW optimizer is used for 3 epochs, with an initial learning rate of $5e-5$ and a batch size of 16. The learning rate is linearly increased in a warm-up period from 0 to $5e-5$ for the first epoch and then linearly decreased to 0. The parameters that resulted in better performance during our empirical analysis are batch_size = 16, warmup_steps=100, nr_epochs=3, and THR = 0.5.

We will refer to this fune-tuned BERT model as *BERT-tuned* and recall that the loss function in it is BCE loss. In our first set of experiments, we evaluate the BERT-tuned model on the GoEmotions and the SemEval-EC datasets and additionally compare it with previous (SOTA) biLSTM models. In the second set of experiments, we evaluate the impact of loss function variations on the performance of BERT-tuned on both datasets. These results show that several variations allow addressing data imbalance better. Building from these results, in the third set of experiments we relate the findings of the hyperparameter optimization, which considers a broader search space over both loss function variations and other hyperparameters.

4.5.1 Comparative Performance Analysis

The top row of Table 4.1 shows the performance of BERT-tuned (with BCE loss) on the GoEmotions and SemEval-EC datasets. On the GoEmotions dataset, where the annotated emotions are more granular (27 emotions), the obtained macro-F1 score is 0.46, which is also what is reported in [48]. For comparison, the SOTA biLSTM model utilized as baseline to compare to BERT in [48] only reaches a macro-F1 score of 0.41. Our biLSTM, presented

earlier in [13], also performs similarly to the baseline biLSTM model presented in [50] and significantly underperforms the BERT-tuned model. Specifically, in a head-to-head comparison over the SemEval-EC dataset, the accuracy increases from 0.85 in the biLSTM model to 0.87 in the BERT-tuned model; the micro-F1 score rises from 0.60 in the biLSTM model to 0.68 in BERT-tuned; the macro-F1 score rises from 0.40 in the biLSTM model to 0.49 in BERT-tuned; and the weighted-F1 score rises from 0.56 in the biLSTM model to 0.65 in BERT-tuned.

4.5.2 Evaluating Variations of Loss Functions in Fine-Tuned BERT

Table 4.1 shows the impact of the 8 loss function variations (shown in the first column) in the performance of BERT-tuned over test datasets for both the GoEmotions and SemEval-EC datasets. We note that, when FL is used, the additional hyperparameters are set as recommended in literature [47]: namely, $\beta = 0.9999$ and $\gamma = 2.0$. We keep these default hyperparameters to fine-tune BERT models with different loss functions in our preliminary evaluation. The results related in Table 4.1 show that many of variations of loss functions result in better micro-, macro-, and weighted-F1 scores than BERT-tuned (with BCE loss). Interestingly, either TWP or CB variations do not improve or even detract from the performance of BERT-tuned (with BCE loss). Performance improves when FL loss is considered over BCE loss. Specifically, FL (over BCE) and its variants (FL-TWP, FL-BC, FL-TPW-CB) result in higher macro-F1 values over both datasets (as well as higher micro-F1 and weighted-F1). The gains in performance are significant over the BERT-tuned (with BCE loss) for both datasets. These results suggest that performance can be further improved if one considers loss function variations, and this prompts us to carry out a broader search over the space of possible hyperparameters.

Table 4.1: BCE refers to BERT-tuned with BCE loss. *-TPW denotes the true positive weights modification to a loss function, *-CB denotes the modification by the class-balance term, and *-TPW-CB denotes the modification by both terms. Highest values along macro-, micro-, and weighted-F1 are highlighted in bold. Abbreviations TrAcc (training accuracy) and TsAcc (testing accuracy) are used in the interest of space.

Setting	GoEmotions					SemEval-EC				
	TrAcc	TsAcc	micro-F1	macro-F1	weighted-F1	TrAcc	TsAcc	micro-F1	macro-F1	weighted-F1
BCE	0.97	0.97	0.54	0.46	0.50	0.90	0.87	0.68	0.49	0.65
BCE-TPW	0.97	0.97	0.54	0.43	0.50	0.90	0.87	0.68	0.50	0.65
BCE-CB	0.97	0.97	0.54	0.44	0.50	0.81	0.82	0.51	0.24	0.40
BCE-TPW-CB	0.97	0.97	0.55	0.47	0.52	0.89	0.87	0.67	0.53	0.65
FL	0.97	0.97	0.57	0.49	0.55	0.91	0.87	**0.69**	0.54	**0.67**
FL-TPW	0.96	0.97	0.57	0.49	0.54	0.91	0.87	**0.69**	0.54	**0.67**
FL-CB	0.97	0.97	**0.58**	**0.51**	**0.56**	0.90	0.87	0.68	**0.55**	0.66
FL-TPW-CB	0.97	0.97	**0.58**	**0.51**	0.55	0.89	0.87	0.68	0.54	0.66

4.5.3 Hyperparameter Optimization Results

As related in Section 4.4, we optimize with respect to macro-F1, and the hyperparameter optimization is carried out over a complex search space defined by the eight loss function variations, the learning rate LR, the classification threshold THR, and the γ, α, and β parameters. Namely: the loss function varies over {BCE, FL}; CB and TPW are Boolean variables $\in [True, False]$; LR $\in [2e-5, 5e-5]$; THR $\in [0.3, 0.5]$; $\gamma \in [0.5, 2.0]$; $\alpha \in [0.5, 2.0]$; $\beta \in [0.99, 0.9999]$; and Random_seed (RS) $\in [0, 100]$ is included as an integer-valued parameter to account for possible divergence due to random initialization.

Fig. 4.2 provides a summary view of the model space by showing the macro-F1 scores of models corresponding to different hyperparameter settings explored via random search over the hyperparameter space outlined above. Fig. 4.2 clearly shows that the FL variations result in better models on each of the datasets. In particular, the FL-CB and FL-TPW-CB variations confer the better-performing models. As we detail below, the best/optimal model is obtained by the FL-CB variation.

We compare the best/optimal model found by hyperparameter optimization to what we refer to as a near-optimal model; the latter is indicated by the preliminary analysis related in Table 4.1. Namely, for the GoEmotions dataset, comparison over loss function variations suggests that FL-CB and FL-TWP-CB result in higher macro-F1 score over other loss function variations. We select BERT-tuned with FL-CB, keeping the default hyperparameters as in BERT-tuned (β, γ, LR, THR), and refer to this model as near-optimal on the GoEmotions dataset. This near-optimal model reaches a micro-F1 score of 0.5775, a macro-F1 score of 0.5091, and a weighted-F1 score of 0.5555 (which are rounded to the second digit after the decimal sign in Table 4.1. Similarly, a near-optimal model on the SemEval-EC dataset is suggested by Table 4.1 to result from the variation FL-CB loss. We refer to this model as near-optimal and recall that it reaches a micro-F1 score of 0.6803, a

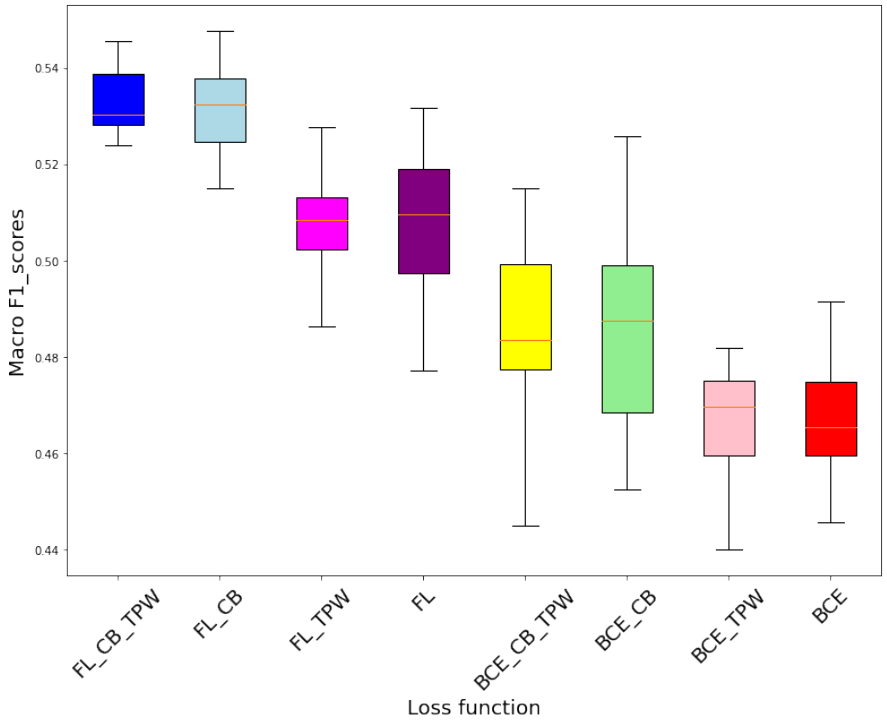
macro-F1 score of 0.5450, and a weighted-F1 score of 0.6646 (rounded to the second digit) shown in Table 4.1. These near-optimal models are compared to the optimal models found via hyperparameter optimization in Table 4.2. The optimal models are significantly better; higher macro-F1 scores (in bold font), as well as higher micro- and weighted-F1 scores.

Table 4.2: Comparison of near-optimal to optimal models on each dataset (GoEmotions and SemEval-EC). The optimal model is the best (highest macro-F1 score) over N models trained over the training dataset, where each model uses hyperparameter values sampled from the hyperparameter space described in Section 4.4. Performance is reported on the testing dataset (accuracy, and F1 scores). Abbreviations TrAcc (training accuracy), TsAcc (testing accuracy), mi-F1 (micro-F1), and w-F1 (weighted-F1) are used here in the interest of space.

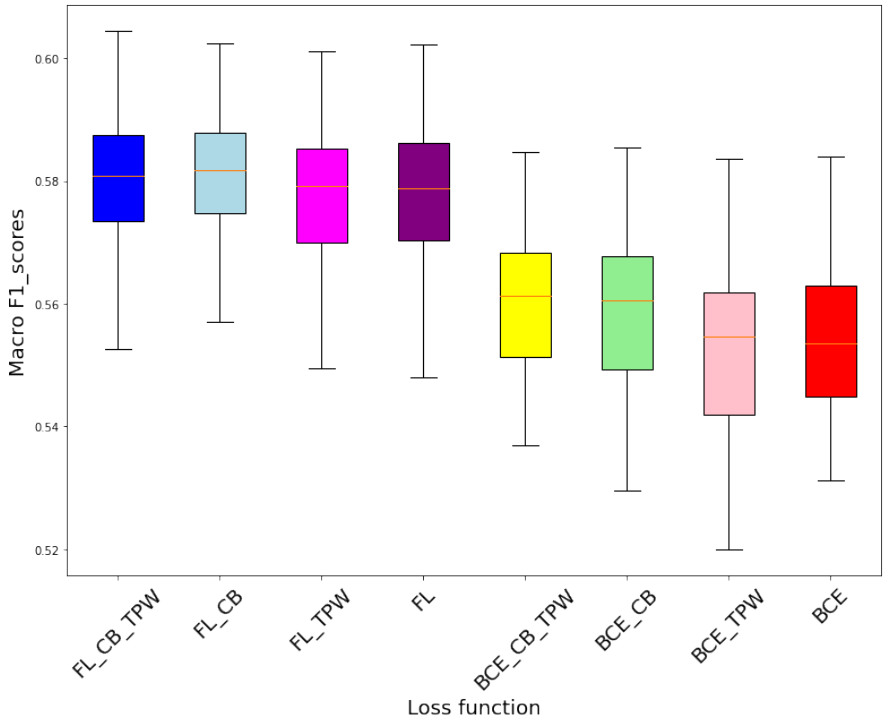
Model Parameters	GoEmotions						
	Loss	α	β	γ	LR	THR	RS
Near-optimal	FL-CB	1.0	0.9999	2.0	$5e-5$	0.5	42
Optimal	FL-CB	1.0344	0.9990	0.9082	2.24E-05	0.3648	26

Model Parameters	SemEval-EC						
	Loss	α	β	γ	LR	THR	RS
Near-optimal	FL-CB	1.0	0.9999	2.0	$5e-5$	0.5	42
Optimal	FL-CB-TPW	1.4779	0.9922	1.5682	$3.77e-05$	0.4097	32

Model Performance	GoEmotions				SemEval-EC			
	TsAcc	mi-F1	macro-F1	w-F1	TsAcc	mi-F1	macro-F1	w-F1
Near-optimal	0.9697	0.5776	0.5091	0.5555	0.8701	0.6804	0.5451	0.6646
Optimal	0.9653	0.6096	**0.5475**	0.6016	0.8608	0.7052	**0.6044**	0.7014



(a) GoEmotions



(b)SemEval-EC

Figure 4.2: Macro-F1 scores of models obtained during hyperparameter optimization for (a) GoEmotions and (b) SemEval-EC datasets.

Table 4.2 provides a class-average view of performance. In Table 4.3, we show the per-class F1 scores obtained by the near-optimal model and the optimal model (the latter as revealed by hyperparameter optimization) over the GoEmotions dataset (top panel) and the SemEval-EC dataset (bottom panel). Table 4.3 shows that hyperparameter optimization has been useful in highlighting models that improve not only overall performance in the presence of data imbalance, but also class-specific performance. Comparison with the BERT-tuned model in [48] in Table 4.3 additionally shows that the optimal model obtained here improves performance in several emotion classes, sometimes doubling macro-F1 score (see 'annoyance').

Table 4.3: Per-class F1 scores achieved by the near-optimal and optimal BERT models over the (top) GoEmotions and (bottom) SemEval-EC dataset.

GoEmotions	**Near-optimal**			**Optimal**			[48]		
Emotion	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
admiration	0.70	0.63	0.67	0.64	0.76	0.70	0.53	0.83	0.65
amusement	0.78	0.84	0.81	0.75	0.91	0.82	0.70	0.94	0.80
anger	0.57	0.43	0.49	0.41	0.57	0.48	0.36	0.66	0.47
annoyance	0.49	0.12	0.19	0.35	0.40	0.37	0.24	0.63	0.34
approval	0.64	0.22	0.33	0.47	0.43	0.45	0.26	0.57	0.36
caring	0.49	0.30	0.38	0.43	0.54	0.48	0.30	0.56	0.39
confusion	0.54	0.35	0.43	0.41	0.51	0.45	0.24	0.76	0.37
curiosity	0.52	0.50	0.51	0.47	0.74	0.57	0.40	0.84	0.54
desire	0.62	0.40	0.49	0.59	0.48	0.53	0.43	0.59	0.49
disappointment	0.56	0.16	0.25	0.36	0.27	0.31	0.19	0.52	0.28
disapproval	0.55	0.26	0.35	0.39	0.47	0.42	0.29	0.61	0.39
disgust	0.58	0.46	0.51	0.48	0.47	0.47	0.34	0.66	0.45
embarrassment	0.64	0.38	0.47	0.62	0.41	0.49	0.39	0.49	0.43
excitement	0.59	0.36	0.45	0.47	0.40	0.43	0.26	0.52	0.34
fear	0.72	0.69	0.71	0.61	0.78	0.69	0.46	0.85	0.60
gratitude	0.94	0.90	0.92	0.91	0.92	0.92	0.79	0.95	0.86
grief	1.00	0.33	0.50	0.50	0.50	0.50	0.00	0.00	0.00
joy	0.66	0.55	0.60	0.59	0.64	0.61	0.39	0.73	0.51
love	0.78	0.80	0.79	0.73	0.88	0.80	0.68	0.92	0.78
nervousness	0.47	0.30	0.37	0.40	0.35	0.37	0.28	0.48	0.35
optimism	0.68	0.45	0.54	0.61	0.58	0.59	0.41	0.69	0.51
pride	0.75	0.38	0.50	0.70	0.44	0.54	0.67	0.25	0.36
realization	0.62	0.14	0.23	0.37	0.18	0.24	0.16	0.29	0.21
relief	0.44	0.36	0.40	0.33	0.45	0.38	0.50	0.09	0.15
remorse	0.61	0.77	0.68	0.57	0.88	0.69	0.53	0.88	0.66
sadness	0.66	0.47	0.55	0.53	0.60	0.56	0.38	0.71	0.49
surprise	0.58	0.48	0.52	0.51	0.56	0.53	0.40	0.66	0.50
neutral	0.72	0.55	0.62	0.63	0.76	0.69	0.56	0.84	0.68
macro-average	0.64	0.45	0.51	0.53	0.57	**0.55**	0.40	0.63	0.46

SemEval-EC	**Near-optimal**			**Optimal**		
Emotion	Precision	Recall	F1	Precision	Recall	F1
anger	0.54	0.42	0.48	0.74	0.82	0.78
anticipation	0.41	0.16	0.23	0.36	0.30	0.33
disgust	0.60	0.42	0.50	0.68	0.82	0.74
fear	0.69	0.73	0.71	0.68	0.78	0.73
joy	0.67	0.57	0.61	0.81	0.86	0.84
love	0.78	0.83	0.81	0.57	0.68	0.62
optimism	0.71	0.46	0.56	0.68	0.82	0.74
pessimism	0.46	0.24	0.31	0.40	0.45	0.42
sadness	0.64	0.51	0.56	0.64	0.76	0.69
surprise	0.60	0.48	0.54	0.34	0.17	0.23
trust	0.22	0.03	0.06	0.23	0.14	0.18
macro-average	0.62	0.50	0.55	0.56	0.57	**0.60**

4.5.4 The Paired Bootstrap Test

In order to compare the performance of different models, we perform a statistical significance test [57]. There are two non-parametric tests commonly used in NLP: approximate randomization [58] and the bootstrap test [59]. The bootstrap test is a paired test, in which we compare two sets and can be applied to compare any metric, such as accuracy, precision, recall or F1 scores. In this work, we apply this test to macro-F1 score to compare the performance of the optimal model with near-optimal model presented in the Table 4.2 and also the performance of the optimal model with the baseline model (non-optimal) presented in the Table 4.2 baseline BERT models.

Focusing on macro-F1 score, we set the following null hypothesis H_0: A does no better than B on the test dataset in terms of the macro-F1 dataset. To evaluate whether we can reject the null hypothesis, we proceed as follows. We first compute $\delta(D)$, which is the performance difference between model A and model B on the test set D. This is used as reference. Then, the bootstrap test starts by repeatedly sampling from the test dataset under the assumption that each sample (of the test dataset D) is representative of the whole population (the whole test dataset D). The bootstrap test creates b samples of the reference dataset D. Each sample contains n instances, which are sampled uniformly at random and with replacement from the reference dataset D. That is, each of the b samples contain n instances drawn at random and with replacement from the test dataset D in our case. The macro-F1 score of model A and model B is computed on each of the samples x^i, where $1 \leq i \leq b$, and the difference $\delta(x^i) =$ macro-F1(A, x^i) $-$ macro-F1(A, x^i) is computed. Every time $\delta(x^i) > 2 \cdot \delta(D)$, an integer s is incremented. The value s/b counts in what percentage of the b samples, the difference in performance between models A and B exceeds (by a factor of two) that on the full test dataset. The value s/b reports on what % of the b samples model A beat expectations and acts as a one-sided empirical p-value. The intuition is that if very few of the samples beat expectations (that is, the p-value is small), then the observed $\delta(D)$ is probably not accidental; hence, p-value is small. We report the

obtained p-values when comparing several models in pairs in Table 4.4.

Table 4.4 shows that the comparisons are significant. In particular, the optimal model outperforms the baseline model (thus, the null hypothesis can be rejected), and this is statistically significant.

4.6 Conclusion

In this chapter we have shown that BERT and specialized loss functions improve performance in EC and are particularly effective to handle data imbalance. Hyperparameter optimization is carried out over a complex hyperparameter search space to reveal a best model. Several directions of research remain. Based on progress in computer vision, we speculate that meta-learning may be just as a powerful tool to further improve multi-label and multi-class EC. Ensemble learning is also an interesting direction, as we show next. Noise, which may be present due to human annotation of datasets, needs additional attention. Integration of data of various modalities may provide a way forward to handle noise. Finally, we anticipate that progress in EC will invariably spur tandem work focusing on regression formulations that can additionally capture the strength of the present emotions.

Table 4.4: The Paired Bootstrap Test for comparing the performance difference between two models: A: Optimal, B: Near-optimal, C: Non-optimal

n = 50	b = 1000		b = 10000	
Dataset	A B	A C	A B	A C
GoEmotions	$\delta(x) = 0.02556$, p-value = 0.175	$\delta(x) = 0.07922$, p-value = 0.003	$\delta(x) = 0.02555$, p-value = 0.1664	$\delta(x) = 0.07922$, p-value = 0.0042
SemEval-EC	$\delta(x) = 0.05261$, p-value = 0.132	$\delta(x) = 0.05921$, p-value = 0.092	$\delta(x) = 0.05261$, p-value = 0.1242	$\delta(x) = 0.05921$, p-value = 0.0976

Chapter 5: Emotion Classification as a Noisy, Multi-class Classification Problem

5.1 Summary

In the previous chapter, we formulated the problem of emotion classification as a multi-label classification problem, essentially allowing for short text to be annotated with multiple emotions. This allowed us to train and evaluate a deep model on the SemEval-EC dataset. However, not all datasets labeled by human annotators consist of instances labeled with multiple emotions. One such dataset is Crowdflower, where each instance is labeled with only one of a set of emotions. So, in this chapter, we broaden our formulation of EC from short, informal text in two forms, to include multi-class classification. As the previous chapter demonstrated, datasets are often imbalanced, and this is certainly the case with Crowdflower, as we detail below. In addition, Crowflower is also very noisy. So, in this chapter we introduce an additional technique to address this issue, by aggregating emotional categories. We evaluate not only the previous models presented in chapters 4-5 in this dataset, but additionally present a stacked ensemble model over binary sub-models that exposes the current state of the art and the remaining challenges in datasets such as Crowdflower.

5.2 Introduction

As already mentioned, one of the challenges with learning to detect emotions in text is the fact that available training data differ in psychological models employed (i.e., annotation schemes) and, more practically, the number of emotions annotated [41]. For instance, the Crowdflower dataset is annotated with 13 different emotions.

Crowdflower is a platform that collects data for developing machine learning algorithms for diverse tasks, including semantic analysis tasks [60]. What we refer to as the Crowdflower dataset here is a dataset made available by the Crowdflower platform in the *Data for Everyone* library [61]. The annotations are based on Ekman's and Plutchik's psychological models, yielding a total of 13 labels: *joy /happiness*, *sadness*, *fear/worry*, *anger*, *surprise*, *enthusiasm*, *fun*, *hate*, *neutral*, *love*, *boredom*, *relief*, *empty*.

Many authors have reported that several emotion classes in the Crowdflower dataset are extremely similar [62]. Repeated efforts to re-label the data have not been successful; for instance, adding more tweets using hashtags such as #happy could only increase accuracy up to 62% [36]. Therefore, in this paper, we merge several emotion classes after removing *empty* and *neutral* and are left with the following 5 emotion classes for the Crowdlower dataset: *joy*, *disgust*, *sadness*, *surprise*, and *anger*. We note that these five classes are the same ones as in [36], with the absence of *sarcasm* and *love* (in [63], the authors consider *fun* instead of *sarcasm*). They are also the five basic emotions over which the three most popular psychological models agree (with surprise replacing fear).

5.3 Methods

5.3.1 Datasets and Data Preprocessing

Details on the Crowdflower dataset is related in Section 5.4.

We consider traditional ML methods here for the sub-learners, such as logistic regression (LR), support vector machine (SVM), random forest (RF), and k-nearest neighbor (KNN) classifiers with various features, and investigate their utility as sub-models via meta-learners. We also experiment with Gradient Boosting classifiers with stacking. We combine these traditional/shallow, feature-based classifiers in a meta-learner via stacking. The process starts with first addressing class imbalance, extracting features from the balanced dataset, training sub-models, and then combining them via stacking, as detailed below.

Imbalanced Data

The Crowdflower dataset is highly imbalanced, as related in Section 5.4. We evaluate two strategies to remedy this, data sampling (under-sampling or over-sampling) and making use of class weight for the loss function. Data sampling methods, such as oversampling using SMOTE, can increase accuracy on the smaller classes but lower the overall performance of a classifier. So, we construct an unbiased train-test split for our classifiers using the stratified sampling technique. We also compute class weights by assigning more emphasis to a class.

Feature Extraction

We utilize features that have been shown useful for sentiment analysis. One of the simplest to map textual data to real-valued vectors is *Bag of Words (BOW). Term Frequency–Inverse Document Frequency (TF-IDF)* features are utilized to find out the terms that are most correlated with each of the labels. We first vectorize tweet texts using TF-IDF and obtain unigrams, bigrams, and trigrams for each class. In this way, training and testing datasets are converted into matrices of TF-IDF features.

Stacking Binary Sub-Models via Stacked Generalization

To better handle class imbalance in the Crowdflower dataset, we train binary sub-models to convert a multi-class classification problem into several binary classification sub-problems. The main idea is that the predictions of the binary sub-models can then be combined to train a meta-learner via ensemble learning. Gradient boosting is fairly robust to overfitting, so after experimenting with different classifiers for sub-models, we finally choose each sub-model to be a Gradient Boosting Classifier. We additionally utilize the MLENS API, which is a Python library for memory-efficient, paralleled ensemble learning, in order to build different number of layers of sub-models to be stacked together.

The sub-models can be combined via *stacking*. Stacking is an ensemble model, where a new model is trained from the combined predictions of two (or more) sub-models. The predictions from the sub-models are used as inputs for each sequential layer and combined

to form a new set of predictions. These can be used on additional layers, or the process can stop with a final result. Ensemble stacking is also often referred to as blending, because all the numbers/predictions are blended to produce a final prediction or classification.

One way to combine the predictions of trained sub-models is via model-averaging [64]. A limitation of this approach is that each model contributes the same amount to the ensemble prediction, regardless of how well the model performs. A variation of this approach, known as a weighted-average ensemble, weighs the contribution of each ensemble member by its trust or expected performance on a holdout dataset. This allows well-performing models to contribute more and less-well-performing models to contribute less to the ensemble prediction. A further generalization of this approach is replacing the linear weighted sum (that is, linear regression) ensemble model that combines the predictions of the sub-models with any learning algorithm. This approach is known as stacked generalization and is often abbreviated as stacking.

Stacked generalization works by deducing the biases of the generalizer(s) with respect to a provided learning set. This deduction proceeds by generalizing in a second space whose inputs are the predictions of the original generalizers when taught with part of the learning set. The generalizer(s) tries to guess the rest of it, as well as guess whose output is the correct prediction. If we view the process as an algorithm, the algorithm takes as input the outputs of trained sub-models and attempts to learn how to best combine the input predictions to make a better output prediction. It is helpful to think of the stacking algorithm as having two levels, level 0 and level 1.

- Level-0: The level-0 data is the training dataset inputs. Level-0 models learn to make predictions from this data.
- Level-1: The level-1 data takes the output of the level-0 models as input. Then, the single level-1 model (which is now a meta-learner), learns to make predictions from this data.

5.4 Results

We first provide some more analysis of the Crowdflower dataset and then relate the performance of the models described in Section 5.3.

5.4.1 Analysis of Crowdflower Dataset

The Crowdflower dataset consists of $39,740$ tweets, where each tweet contains one of 13 labels listed above. We merge several emotions with highest correlations (these contribute to the most classification error) and obtain a total of 5 emotions. The class distribution before and after aggregation is shown in Figure 5.1. The breakdown is shown in Table 5.1. After preprocessing, as described in Section 5.3, an $81:9:10\%$ split is used to obtain the training, validation, and testing datasets.

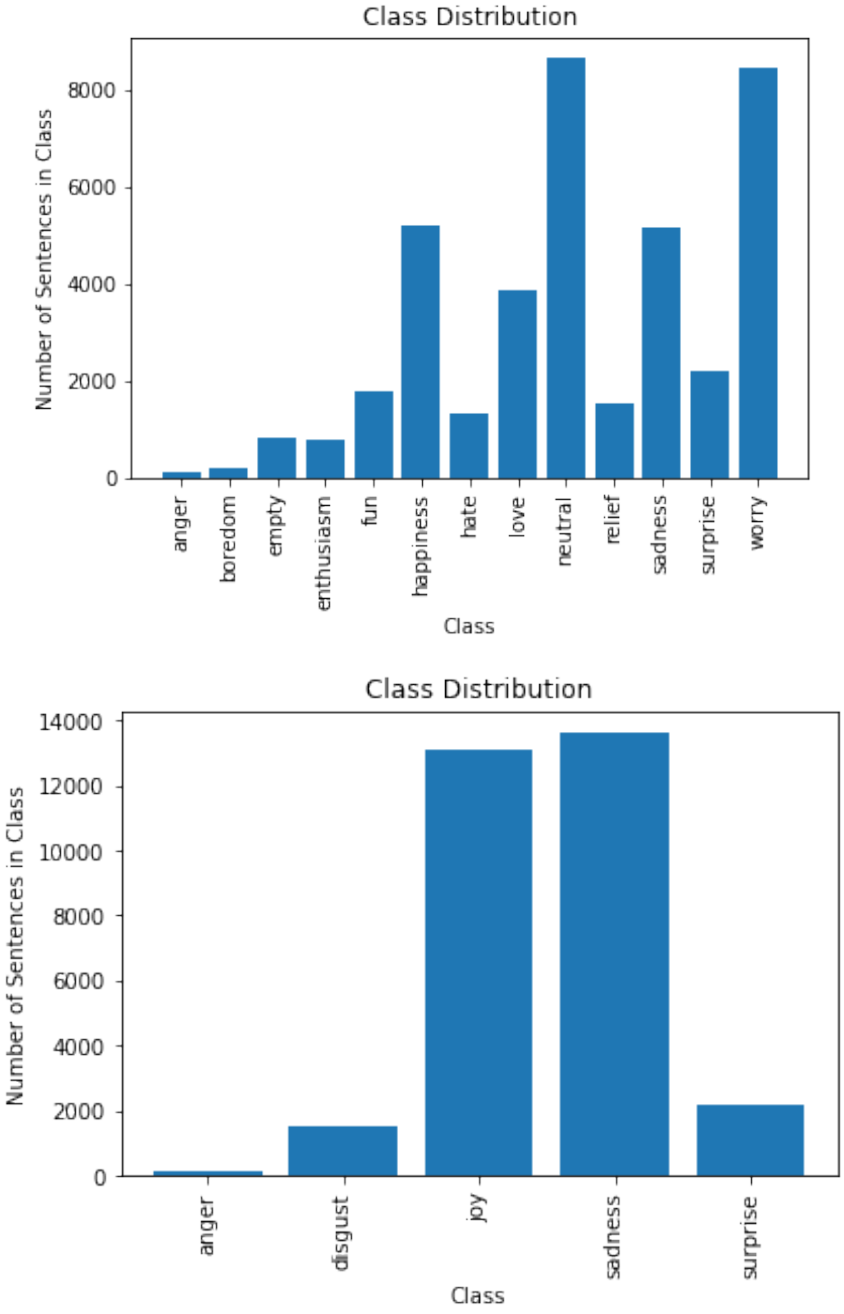


Figure 5.1: Class distribution (top) before and (bottom) after aggregation in the Crowdflower dataset.

As Figure-5.1 shows, the Crowdflower dataset is highly imbalanced, with a difference of close to a factor of 80 between some classes (110 instances for *anger* versus 8459 for *worry*). The Crowdflower dataset is also comparably noisy (to SemEval-EC), as it is annotated via crowdsourcing; several classes are not labeled correctly. For instance, for 'priscilla', and 'translate', the most correlated bigrams are 'awesome happy' and 'waiting friend,' and the most correlated trigrams are 'happy star wars' and 'today going long'.

Table 5.1: Class sizes in Crowdflower dataset.

sentiment	**# tweets**
anger	110
boredom	179
enthusiasm	759
fun	1776
happiness	5209
hate	1323
love	3842
neutral	8638
relief	1526
sadness	5165
surprise	2187
worry	8459
empty	567

5.4.2 Stacking of Binary Sub-models for Multi-Class Emotion Classification on Crowdflower Dataset

The accuracy of each sub-model (trained over the Crowdflower dataset) is shown in Table 5.2. Each sub-model is trained in a binary setting. For instance, the cls-joy model is trained to predict joy as the positive class, with all other classes merged into the negative class. Table 5.2 shows better performance from the sub-models trained to predict disgust, surprise, and anger. Each sub-model is a Gradient Boosting Classifier. The number of boosting stages to perform, n_estimators, is set to 10. and learning_rate is set to 1.

Table 5.2: Comparison of the performance of various binary sub-models on the Crowdflower dataset.

Sub-model binary classifier	**Performance (accuracy)**
joy	0.67
disgust	0.96
sadness	0.59
surprise	0.93
anger	0.99

As described in Section 5.3, the predictions of each binary sub-model over the training examples comprise the training dataset for the meta-learner. Predictions are stacked together to build train/test sets for an LR meta-learner. We also perform 10-fold cross-validation for training using the LBFGS optimizer. The stacked test accuracy obtained is 56.1%. Based on the LR prediction probabilities, we can choose the highest-probability emotion as a single prediction or choose multiple emotions to obtain a multi-emotion prediction.

The MLENS library is also utilized for training as a paralleled ensemble learning. For the first layer (layer-0), we choose the following classifiers to train the sub-models: RF (n_estimators=20); SVM (kernel='rbf'; LR (class_weight ='balanced'); and KNN (n_neighbors=7).

In the next layer of ensemble (layer-1), we use LR as the meta learner in which by varying the sub-models, we obtain the performances reported in Table 5.3. The evaluation metrics reported in this table are Score-m (mean test_score), Score-s (test_score standard deviation), Ft-m (mean fit_time), Ft-s (standard deviation fit_time), Pt-m (mean prediction_time), and Pt-s (standard deviation prediction_time). Both methods, the stacked ensemble and the paralleled ensemble, show comparatively similar performance in terms of test set accuracy.

Table 5.3: Performance of the level-1 LR meta-learner over varying level-0 classifiers is related along several metrics. The MLENS API is employed for memory-efficient paralleled ensemble learning.

Layer-1	Layer-0 Classifier	Score-m	Score-s	Ft-m	Ft-s	Pt-m	Pt-s
LR	KNN	0.51	0.05	0.73	0.07	2.57	0.12
LR	LR	0.56	0.00	0.99	0.00	0.00	0.00
LR	RF	0.56	0.00	0.62	0.40	0.04	0.00
LR	SVM	0.56	0.00	3.90	0.02	2.27	0.01
LR	LR	Prediction score: 0.561					

5.4.3 Parameters of Deep NN Architectures

We now summarize our results when employing the deep NN architectures described in Section 5.3. Parameters for learning are as follows:

- NB_WORDS = 30000 (Parameter indicating the number of words in the dictionary)
- VAL_SIZE = 1000 (Size of validation set)
- NB_EPOCHS = 50 (Number of epochs)
- BATCH_SIZE = 512 (Size of the batches used in the mini-batch gradient descent)
- MAX_LEN = 100 (Maximum number of words in a sequence)
- GLOVE_DIM = 100 (Number of dimensions of the GloVe word embeddings)
- MAX_SENT_LEN = 300 (character-based length)
- MAX_DOC_LEN=5 (Number of sentences in text)

Multi-class Classification on Crowdflower Dataset

We now apply the deep NN architectures introduced in chapter 4 to the Crowdflower dataset.

For the task of multi-class emotion classification on the Crowdflower dataset, all the deep NN models are trained by choosing softmax as classification function and categorical cross_entropy as the loss function for the last layer. We use MaxPooling to downsample and dropout of 0.2 to avoid overfitting (the scenario when accuracy is high on the training dataset but low on the the testing dataset.

The performance of the deep NN models on the Crowdflower dataset is related in Table 5.4. As Table 5.4 shows all models except multi-channel, multi-filter CNN-BiLSTM suffer from overfitting the training dataset; in other words, their accuracy on the training is above 97%, but their accuracy on the testing dataset is below 60%. In contrast, the multi-channel, multi-filter CNN-BiLSTM model achieves training and testing dataset of about 69% and 64%, respectively. The macro-F1 score and weighed-F1 score of this model are also the same or better than other models; 0.30 and 0.60, respectively. Taken together, this comparative analysis indicates that the multi-channel, multi-filter CNN-BiLSTM model is superior over the other four models for the task of multi-class emotion classification on the Crowdflower dataset.

Table 5.4: Comparison of performance of deep NN models on the Crowdflower dataset. The highest accuracy on the testing dataset is highlighted in bold font.

CNN model	Train accuracy	Test accuracy	macro-F1	weighted-F1
Multi-filter CNN	0.987	0.581	0.30	0.59
Static, Multi-filter CNN with GloVe	0.978	0.570	0.29	0.56
Non-Static, Multi-filter CNN with GloVe	0.984	0.591	0.30	0.57
Multi-channel, Multi-filter CNN-BiLSTM	0.692	**0.639**	0.30	0.60
Multi-channel, Multi-filter CNN (Kim-CNN)	0.984	0.585	0.31	0.57

5.5 Conclusion

In this chapter, we have approached emotion classification as a multi-class problem, employing popular psychology models of basic and advanced human emotions. We have presented various methods, including a stacked ensemble model over binary sub-models and several deep neural network architectures. The CNN-BiLSTM model achieves 64% accuracy on the multi-class Crowdflower dataset. This is higher than other deep models, but nowhere near the 85% accuracy on the multi-label SemEval-EC dataset presented in an earlier chapter. Some of the reasons for this can be attributed to the deep imbalance in the Crowdflower dataset. However, the primary reason is noise due to incorrect labeling, an issue that has been reported by various researchers on this dataset. Nonetheless, the stacked generalization approach we present here achieves good per-class F1 scores, which suggests that the approach, combined with aggregation of emotion classes, is effective at addressing data imbalance.

Stacked generalization is a form of ensemble learning, which is a powerful machine learning approach under the umbrella of meta-learning. Specifically, as demonstrated in this chapter, stacking uses another machine learning model, a meta-model, to learn how to best combine predictions of possibly unreliable contributing ensemble members. This line of work now guides us more broadly into the area of meta-learning via the concept of ensembles, which we pursue and evaluate next for emotion classification.

Chapter 6: Ensemble Learning for Emotion Classification

6.1 Summary

The previous chapter utilized hyperoptimization to identify a best/optimal BERT-based model to tackle the issue of data imbalance in the presence of fine-grained emotion categories. A detailed analysis of the models explored and compared during hyperoptimization showed high model variance even over models utilizing the same loss function (or variation of). In this chapter, our objective is to overcome the high variance of fine-tuned BERT models. To do so, we propose novel ensemble methods over BERT sub-models. This approach falls under the umbrella of ensemble learning, which is a technique that combines multiple machine learning algorithms to produce one optimal predictive model with reduced variance. The experiments we relate in this chapter show a carefully thought-out ensemble architecture can boost the performance up to 88.24% accuracy. Comprehensive experiments show that the ensemble model outperforms the existing baseline models and achieves micro-, macro-, and weighted- F1 scores of 72%, 58%, and 70%, respectively.

Chapter 7: Conclusion

The work presented in this dissertation addresses fundamental, basic research challenges that promise to advance our quest towards social machines capable of understanding us and utilizing this understanding to help us understand and respond to one another in productive ways. Moving closer, however, to sociable machines will require integrating mature, reliable AI platforms that generalize well and have the ability to keep improving. In this chapter we articulate some primary avenues we identify to further advance machine understanding of human behavior and interactions in social media platforms.

7.1 Accountable Machines

As the work presented in this dissertation makes clear, the level of sophistication we can now encode in machine learning models keeps increasing. This complexity, while allowing us to capture the knowledge embedded in data, comes at a cost. The deep models we present here are not accountable. That is, if we are curious to understand how they arrive at certain decisions, we are out of luck. At a fundamental level, if we were to try to understand for instance, why any of our models have made a decision that "I was so happy to see that" expresses the emotion of relief rather than happy, we would have to embark on a new line of research to peek into the model and obtain an answer we can summarize and make sense of. More broadly, this challenge goes by several names in machine learning and AI research, such as interpretability, explainability, and accountability [65, 66].

Model accountability is not just a nice feature to have. It may have important repercussions if, for instance, the models presented here and others are employed to track the emotional well-being of employees or students. Research on how to equip models with at least the ability to help us understand how decisions are reached, is highly active. One

approach is via attention mechanisms [67]. These mechanisms are a way to non-uniformly weight the contributions of various input features and so can provide a partial answer to, say, what words were most important and influenced the prediction of a certain emotion or a set of emotions. We expect that, as research in EC matures, more explainability mechanisms may be pursued, integrated, and evaluated in this domain.

7.2 Machines with Linguistic knowledge

Current deep learning approaches for sentiment or emotion recognition, including the ones presented in this dissertation, do not fully exploit sentiment linguistic knowledge. We suspect that fully employing linguistic resources will advance research in this domain. One could integrate three kinds of emotion linguistic knowledge, such as *emotion-word lexicon*, *negating sentimental words*, and *NRC emotion intensity word lexicon* [68]. Emotion intensity and word-emotion association lexicons were introduced in Chapter 3. Besides such lexicons, one could also leverage negation words or negation cues such as 'never', 'not', and also intensity words or intensifiers, such as 'very', 'a lot', 'absolutely'.

In particular, by using an emotion intensity lexicon, one can add a prior distribution for a specific primary emotion for each word to a model, which can be useful in determining the emotion intensity degree (polarity) of longer texts, such as phrases and sentences. A few rows of NRC Emotion/ Affect Intensity Lexicon are shown in Figure A.1. As Figure A.1 shows, for example, the word 'outraged' offers a 0.964 prior probability to a position which includes this word. For the word 'worry' the probabilities are anticipation = 0.492, fear = 0.578, sadness = 0.641. The emotion intensity of a phrase shows the strength of associated emotion within a sentence. An intensity word in a sentence can change the valence degree of an emotion word in a text or increase the emotion intensity of a text. Negation cues may be important, as well, as they reverse or increase the affect of each emotion word.

	word	emotion	emotion-intensity-score
0	outraged	anger	0.964
1	brutality	anger	0.959
2	hatred	anger	0.953
3	hateful	anger	0.940
4	terrorize	anger	0.939
5	infuriated	anger	0.938
6	violently	anger	0.938
7	furious	anger	0.929
8	enraged	anger	0.927
9	furiously	anger	0.927

Figure 7.1: Emotion intensity lexicon.

7.3 Multi-Modality for Emotion Recognition

The work presented in this dissertation presents, to some extent, the best that machines can do with the (text) information given. This prompts a fundamental question: Is language enough? back our motivating question of what emotion(s) is/are present in the sentence "I was so happy to see that." Is text enough? It will be duly noted that a new language has evolved over the past years since social media has become a pervasive platform of communication: the language of emojis. Emojis are powerful. Among other roles, they help disambiguate text. In the same manner that they help humans hone in on the right emotion, they ought to help machines do so. In fact, much research in machine learning has considered both text and emojis or emoticons to disambiguate text and thus better detect emotions. This is an example of leveraging data of multiple modalities (text versus emojis). While very promising, it is also limiting. Many datasets do not come with both emojis and emotion labels.

However, leveraging multi-modal data presents opportunities. Multi-modal analysis has emerged as a new domain at the intersection of NLP, computer vision, and speech processing. The idea is to leverage the diversified information, (e.g., textual, acoustic and

visual), for learning a model with the objective of improving the overall performance of the model. Images, video, emojis, speech all present additional sources of information that can be leveraged to help machines better understand us. However, while promising, the bottleneck to advancing this line of research is the availability of benchmark datasets to allow training models that integrate all these sources of information. However, the rapid-pace advancements in computer vision suggest that integrating images and video to text will push forward emotion recognition and even open the way for machines to accomplish more complex tasks with the information streaming in social media platforms. One such task can be broadening the formulation beyond classification to regresssion; that is, predicting not just which emotions are present, but the intensity at which they are present. This formulation provides richer and more nuanced information.

7.4 Transfer Learning to Improve Generalizability

So far, we have developed models, trained them on a training subset of a specific dataset, and then evaluated them on a held-out, testing subset of that same specific dataset. If one thinks of deploying models in an actual platform, to support specific tasks, how likely it is that a model trained on a curated, labeled dataset, will do a good job on new instances? This is a fundamental question in machine learning. It relates to the generalizability of a model. When we train a particular machine learning model, we don't just want it to learn to model the training data. We want it to generalize to data it has not seen before.

Models trained on one dataset and tested on another, entirely different dataset, generally do not perform well. Datasets have different distributions and intrinsic characteristics. So, how can we improve their generalizibility? One way forward is via transfer learning. Transfer learning is a machine learning method, where a model developed for a specific task A is reused as the starting point for a model on a second task B. Transfer learning improves learning in a new task through the transfer of knowledge from a related task that has already been learned. The main challenge to embarking on this avenue at the moment is the variation of annotation schema over benchmark datasets. As we have noted several

times throghout this dissertation, the three main datasets that we have employed and have become the standard in the community, SemEval-EC, Crowdflower (to some extent, though noise in it makes it less ideal for training reliable models), and GoEmotions have different granularity of emotions and even different emotions. We speculate that more datasets with the same annotation schema, or strategies to translate or aggregate annotation schema will be important lines of research and help us mature these models and make them ready for deployment.

Appendix A: Text Processing Preliminaries

In this chapter, we will discuss about the basic steps of a conventional text classification framework which consists of text normalization and pre-processing, feature extraction and text representations, and classification stages. Readers already familiar with these basic NLP concepts are welcome to skip ahead.

A.1 Text Pre-processing

The goal of text pre-processing is to transfer text from human language to a machine-readable format for further processing by an algorithm. This is an important step before any classification or regression task; the unstructured nature of data like text does not lend itself to machine understanding. So, more pertinently, our goal is convert a chunk of text to a list of clean tokens that can then be leveraged for text mining and/or NLP tasks.

We first provide an overview of simple text pre-processing techniques. Some steps of text cleaning are task-specific, but most are considered common enough that they are part of the typical workflow in any NLP task. We start first with *text normalization.*

A.1.1 Text Normalization

Noise removal is the first step. This is defined as cleaning text from any unwanted symbols, removing metadata or markups and finally extracting valuable data from formats such as JSON. The specific steps of text cleaning tend to be more task-specific and include:

1. **Removing Stop Words:** "Stop words" are the most common words in a language like "the", "a", "on", "is", "all". These words do not carry important meaning and are usually removed from texts.

2. **Removal of URLs and Punctuation:** HTML tags and URLs must be deleted from text. We also must remove emojis/ emoticons, or convert them into words.

Punctuation, frequent symbols, words, rare or repetitive-letter words must also be removed from texts.

3. **Lowercasing:** We must convert all words into one case, either loweror uppercase.

Stemming

Stemming is the process of reducing inflection in words into their stem (root) forms (e.g. 'troubled', 'troubles' converts to stem forms 'trouble'). The 'root' in this case may not be a real root word, but just a canonical form of the original word. It might be a mapping of a group of words to the same stem even if the stem itself is not a valid word in the language.

Stemming uses a crude heuristic process that removes the ends of words in order to correctly convert words into its stem form. So the words "trouble", "troubled" and "troubles" might actually be converted to "troubl" instead of trouble because the ends are removed.

The main algorithms used for this purpose are the *Porter Stemming* algorithm [69], which removes common morphological and inflexional endings from words, the *Snowball (Porter 2)* algorithm, and the *Lancaster Stemming* algorithm. Among these three common algorithms, Lancaster is a very aggressive stemming algorithm. The stemmed representations for Porter and Snowball algorithms are usually fairly intuitive to a reader; not so with Lancaster, as many shorter words become totally obfuscated.

Lemmatization

This is the process of converting a word to its base form. The difference between stemming and lemmatization is as follows: Lemmatization considers the context and converts the word to its meaningful base form, whereas stemming just removes the last few characters, often leading to incorrect meanings and spelling errors. For example, lemmatization would correctly identify the base form of 'caring' to 'care', whereas, stemming would remove the 'ing' part and convert it to car. Sometimes, the same word can have multiple different 'lemma's. So, based on the context used, one should try to identify the 'part-of-speech' (POS) tag for the word in that specific context and extract the appropriate lemma.

Expanding Contraction

Contractions are shortened versions of words or syllables. Contractions of words are created by removing specific letters and sounds in either written or spoken forms in the English language, specially in informal language. In the case of English contractions, they are often created by removing one of the vowels from the word. For example, the words "do not" are shortened to "don't" and "I would" to "I'd". We should convert each contraction to its expanded, original form to normalize text.

NLP Libraries

We can carry out text processing by utilizing the Natural Language Toolkit (NLTK) which is a suite of libraries and programs for symbolic and statistical natural language processing. The most standard Python libraries for text processing in English are:

1. NLTK Library.
2. Beautiful Soup Python library for pulling data out of HTML and XML files.
3. Keras API's dataset preprocessing utilities
4. Inflect toolkit which properly generates plural form, singular nouns, ordinals, indefinite articles.
5. pycontractions which is a python library for expanding and creating common English contractions in text.

A.1.2 Tokenization

Tokenization is the process of splitting the given text into smaller pieces called tokens. It is a fundamental step in both traditional NLP methods such as Count Vectorizer and Advanced Deep Learning-based architectures like Transformers. Tokenization is performed on large chunks of text to obtain tokens at different levels, i.e. corpus are tokenized to sentences, sentences are tokenized to words and words can be tokenized to characters.

After tokenization, we have a vocabulary which is constructed by a set of unique tokens. The following tokens are then used to prepare a vocabulary. Vocabulary refers to the set of unique tokens in the corpus.

A.2 Vector Semantics

Words have meanings and context has an important role in defining the meaning of words. Words that occur in similar contexts tend to have similar meanings. This link between similarity in how words are distributed and similarity in what they mean is called the distributional hypothesis which was first proposed by linguistic in [70]. He notices that words like 'eye' and 'examined' tend to occur in the same environment. The idea of vector semantics originated from this hypothesis by learning representations of the meaning of words, called embeddings, directly from their distributions in texts. These representations are applied widely in every natural language processing application that makes use of meaning, and form the foundation of the more powerful contextualized word representations such as ELMo [71] and BERT [72] that will be introduced later in this chapter.

The idea of vector semantics is to represent a word as a point in some multi-dimensional semantic space. Vectors for representing words are generally called embeddings, because the word is embedded in a particular vector space. Based on definition of philosopher Ludwig Wittgenstein, "the meaning of a word is its use in the language", instead of using some logical language to define each word, we should define words by some representation of how the word was used by actual people in speaking and understanding. Therefore, two words that occur in very similar distributions (that occur together with very similar words) are likely to have the same meaning, i.e., the co-occurrence of words in a corpus can teach us something about its meaning. Sometimes, it means they are similar or sometimes it means they are opposite.

A.3 Feature Learning

Machine learning algorithms cannot work with raw text directly, therefore, text must be converted into vectors of numbers. Within traditional word embeddings, two categories are widely used:

- Frequency based Embeddings
- Prediction based Embedding

In this Chapter, we will first introduce frequency-based embedding such as TF-IDF and then will introduce prediction based embeddings such as Word2Vec.

A.3.1 Bag-of-Words

Bag-of-words (BoW) is a method for extracting features from text for use in machine learning modeling. A bag-of-words model is a representation of text that describes the occurrence of words within a document. It involves two steps: We first build a vocabulary of known words. Then, we measure the presence of known words within text. For example, consider the following text corpus (which is a large and unstructured set of texts): *"This is the first document. This document is the second document. And this is the third one. Is this the first document?"*

The vocabulary of model comprising of unique words is as follows: 'and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this' Finally, binary vector for the first sentences in corpus is the following: [011100101]

Count vectorization: Count vectorization is a method based on counting the number of occurrences of each word appearing in a document. For example, for the previous example, we have [020101101] for the second sentence.

A.3.2 TF-IDF

An alternative to calculating word frequencies is TF-IDF (Term Frequency – Inverse Document Frequency) which is a scoring measure widely used in information retrieval (IR) or

summarization. TF-IDF is intended to reflect how relevant a term is in a given document and highlights words that are more frequent in a document but not across documents based on co-occurrence matrix. A common deficiency of TF-IDF vectors is that they are long and sparse.

- **Term Document:** how often a given word appears within a document.
- **Inverse Document Frequency:** downscales words that appear frequently across documents.

A.3.3 Word Embeddings

Word embedding is a powerful technique for language modelling and feature learning which transforms words in a vocabulary to vectors of continuous real numbers. The technique normally involves a mathematic embedding from a high-dimensional sparse vector space (e.g., one-hot encoding vector space, in which each word takes a dimension) to a lower-dimensional dense vector space. Each dimension of the embedding vector represents a latent feature of a word. The vectors may encode linguistic regularities and patterns. The learning of word embeddings can be done using neural networks or matrix factorization. In the following sections, we introduce several commonly-used and state-of-the are word embeddings.

For example, the following visualizations of embeddings show geometrical relationships that capture semantic relations like the relation between a country and its capital:

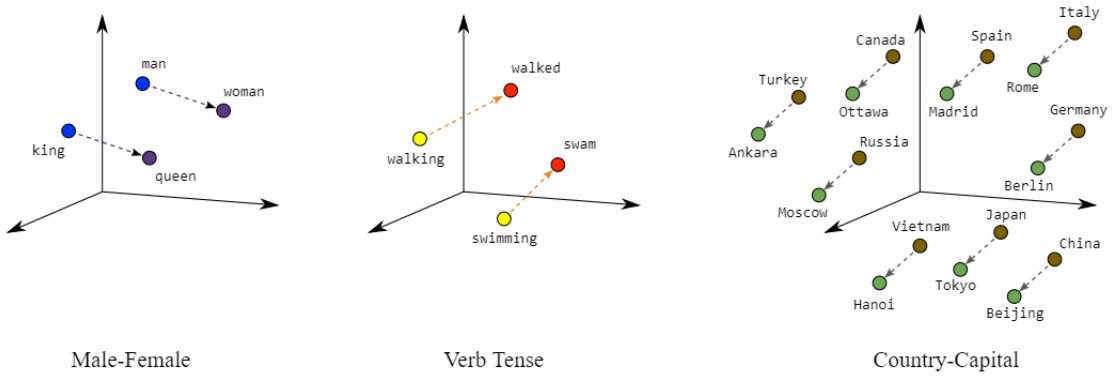


Figure A.1: Analogies of embedding vectors.

A.3.4 Word2Vec

One commonly used word embedding system is *Word2Vec*, which is essentially a computationally-efficient neural network prediction model that learns word embeddings from text. The main idea is that, instead of representing words by long and sparse vectors, we can have words with similar context occupy close spatial positions by the use of vectors that are short and dense. Word2Vec is an algorithm based on the idea of distributed representations. Intuitively, Word2vec introduces some dependence of one word on the other words to construct such an embedding. It was developed by Google and was trained on 100 billion words on Google News dataset.

Word2Vec is classified under two approaches both involving shallow neural networks: Skip-Gram and Common Bag Of Words (CBOW).

The idea behind Word2Vec is that instead of counting how many times each word w occurs near word 'apricot' we can train a classifier on a binary prediction task: "Is word w likely to show up near apricot?" Based on the context window, which is defined as the number of words appearing on the left and right of a target word, we have:

CBOW (Continuous Bag Of Words) Model: This method takes the context of each word as the input and tries to predict the target word corresponding to the context.

Skip-Gram: Given a word, this method predict its neighbors (i.e., its context).

Word2Vec algorithms like Skip-Gram are a popular and efficient way to compute dense embeddings. Skip-Gram trains a logistic regression classifier to compute the probability that two words are *likely to occur nearby in text*. This probability is computed from the dot product between the embeddings for two words. Stochastic gradient descent is used to train the classifier and so learn embeddings that have a high dot product with embeddings of words that occur nearby and a low dot product with noise words. The skip-gram architecture is shown in Figure A.2.

CBOW, on the other hand, predicts the probability of a word given a context. A context may be a single word or a group of words depending on the window size. The architecture of CBOW in shown in Figure A.3.

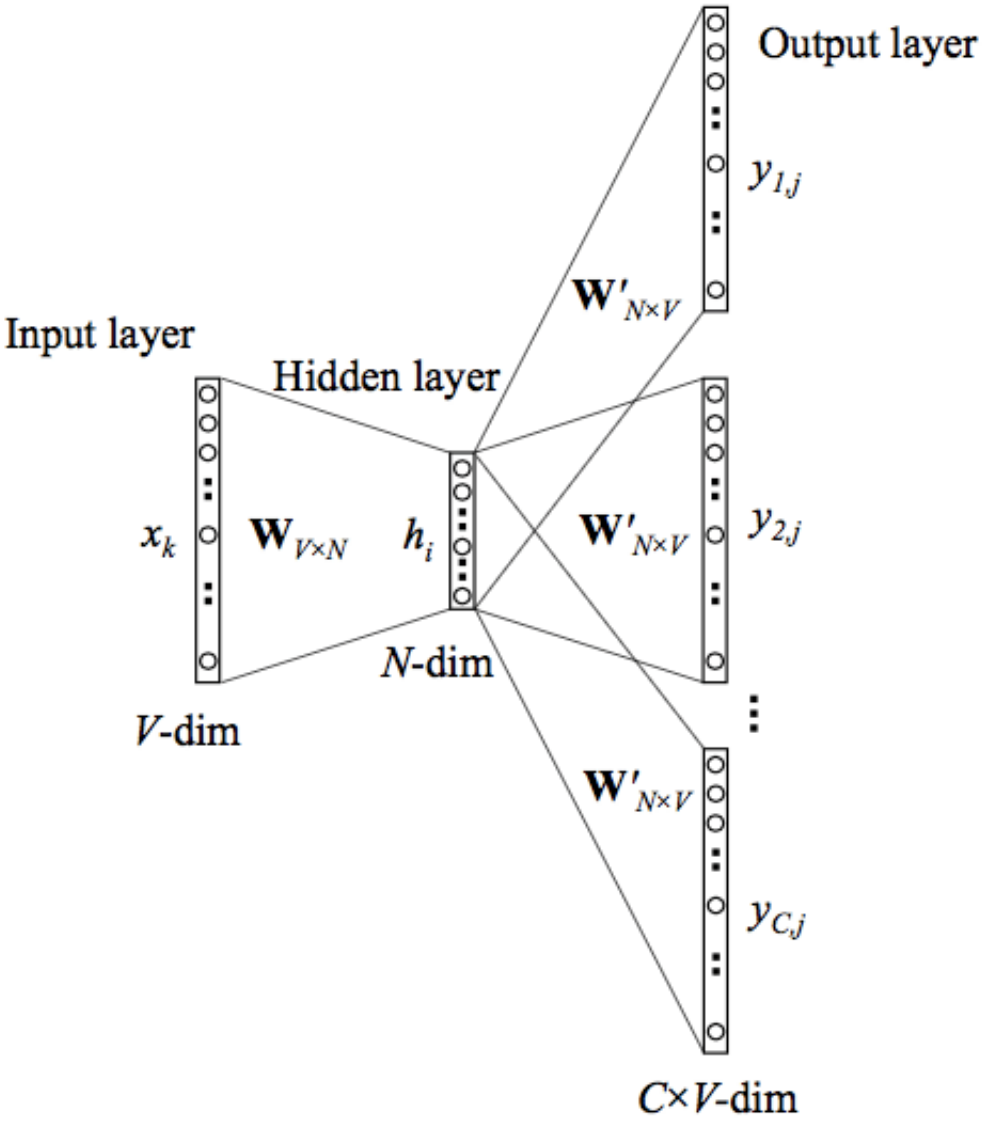


Figure A.2: Skip-Gram architecture.

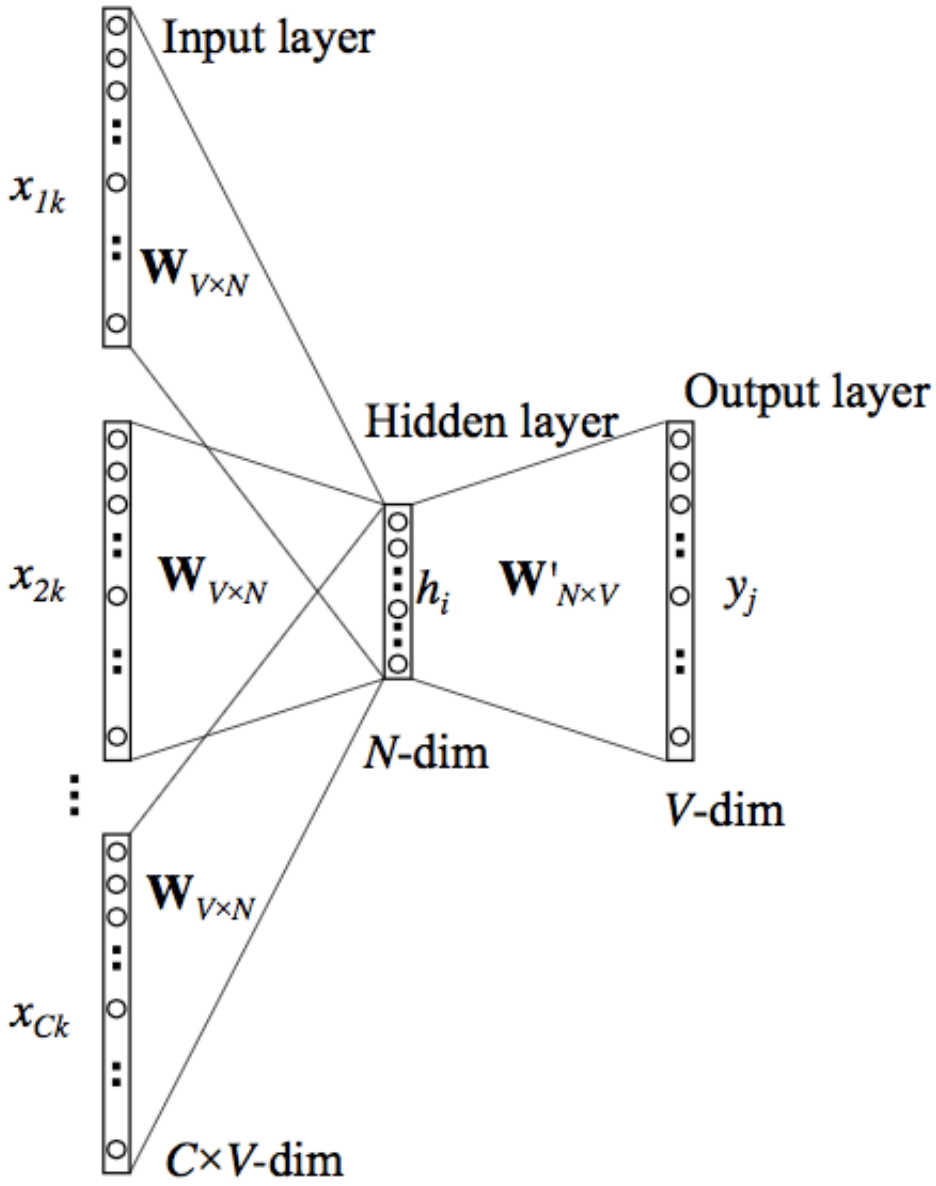


Figure A.3: CBOW architecture.

A.3.5 GloVe

GloVe is a widely-used embedding model besides Word2Vec [73]. The idea of GloVe word embedding is that it derive the relationship between the words from global statistics. GloVe

is based on ratios of probabilities from the word-word co-occurrence matrix, combining the intuitions of count-based models while also capturing the linear structures used by methods like word2vec. A co-occurrence matrix 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. GloVe leverages both Global matrix factorization methods like latent semantic analysis (LSA) for generating low-dimensional word representations Local context window methods such as the skip-gram model.

A.3.6 FastText

One major drawback for word embedding methods such as Word2Vec and GloVe is their inability to deal with unkown out-of-vocabulary (OOV) words. These embedding techniques treat word as the smallest unit and try to learn their embedding vector. Hence, Word2Vec or GloVe methods can not retrieve a representation if a word does not appear in the corpus. FastText was developed by Facebook Research and has shown excellent results on many NLP problems, such as semantic similarity detection and text classification [74]. FastText is able to achieve better performance for word representations and sentence classification, in the case of rare words, by making use of character level information. Each word is represented as a bag of character n-grams in addition to the word itself to deal with this problem. In fact, instead of learning vectors for words directly like GloVe or Word2Vec, FastText represents each word as an n-gram of characters. For example, for word 'artificial' with n=3 as the number of n-grams for representation, fastText representation is:

$\langle$ar, art, rti, tif, ifi, fic, ici, ial, al$\rangle$

where the angular brackets indicate the beginning and end of the word. FastText provides two models for word representations like Word2Vec: Skip-Gram and CBOW:

As explained earlier in Word2Vec, the Skip-Gram model learns to predict a target word using context words. On the other hand, the CBOW model predicts the target word using its context. The context is represented as a bag of the words contained in a fixed size window around the target word. In figure A.4, the difference between two models are depicted. As

an example, suppose we have the following sentence:

'Poets have been mysteriously silent on the subject of cheese' and the target word is 'silent'

- The Skip-Gram model predicts the target using a random close-by word, like 'subject' or 'mysteriously'.
- The CBOW model takes all the words in a surrounding window (window of size = 4: (been, mysteriously, on, the)), and uses the sum of their vectors to predict the target.

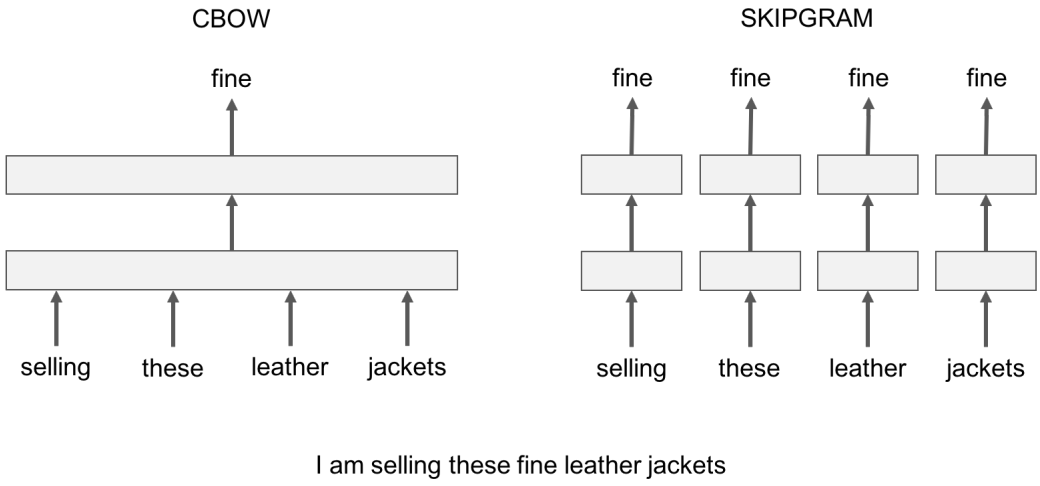


Figure A.4: FastText: Skip-Gram vs CBOW.

A.4 Contextualized Word Representations

There are several challenges with previous traditional word representations:

- Learning high quality representations can be challenging.
- Modeling complex characteristics of word use such as syntax and semantics is complicated.

- Linguistic contexts change word use, in other words, word meaning change across linguistic contexts (polysemy).

Because of these challenges, several representations were proposed that each word embedding is a function of the entire input sentence [71]. In the following chapters, we introduce several of such word representations such as such as BERT, ELMo, and GPT-2 by which incorporating contextulized embeddings into word embeddings has shown significant improvement in text classification.

A.4.1 Sequence-to-Sequence Models

In this section, we will introduce encoder-decoder networks, or sequence-to-sequence models. These models have many applications and been used in machine translation, summarization, question answering, and dialogue modeling. and can generate variable length and contextually aware output sequences.

The key idea underlying these networks is the use of an encoder network which takes an input sequence and creates a contextualized representation of it. The final hidden state of the encoder, h_n, serves as the context of input sequence and is passed to the decoder which generates a task-specific output sequence. The encoder and decoder networks are typically implemented with the same architecture, often using recurrent networks such as RNN. In Figure A.5, a simple architecture of an RNN-based encoder decoder is applicable to machine translation tasks.

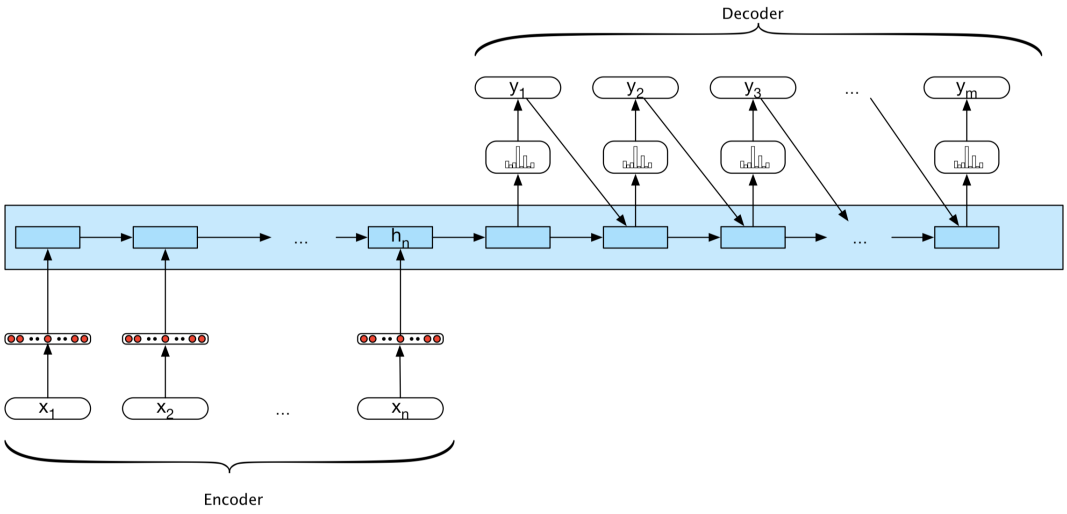


Figure A.5: RNN-based encoder-decoder architecture [57].

Generally, we can leverage any of simple RNNs, LSTMs, GRUs, or convolutional networks to serve as encoders. Also, instead of using one single layer of encoder, we also can build stacked layers of encoders where the output states from the top layer of the stack are taken as the final representation.

Encoder provides the contextualized representation for decoder by its final hidden state. Decoder which is typically an LSTM or GRU-based RNN, autoregressively produces an output sequence. A sequence is generated when each hidden state is conditioned on the previous hidden state and also the output generated in the previous state. The simple formula as following demonstrates the process of encoding-decoding in sequence to sequence modeling:

$$h_0^d = c$$

$$h_t^d = g(y^{t-1}, h_{t-1}^d)$$

$$z_t = f(h_t^d)$$

$$y_t = \text{softmax}(z_t)$$

Google Translate started using such a model in production in late 2016. These models

are explained in [75].

A.4.2 Attention Mechanisms [1]

Bidirectional RNN and LSTM are capable of dealing with temporal dependencies in data but not on very long-range terms. In practice, the long-range dependencies are still problematic to handle. Thus, a technique called the Attention Mechanism was proposed. This mechanism overcomes the deficiencies of these simple approaches to context by taking the entire encoder context into account, and dynamically updating during the course of decoding [57]. The attention mechanism in neural networks is inspired by the visual attention mechanism found in humans. That is, the human visual attention is able to focus on a certain region of an image with "high resolution" while perceiving the surrounding image in "low resolution" and then adjusting the focal point over time. In NLP, the attention mechanism allows the model to learn what to attend to based on the input text and what it has produced so far, rather than encoding the full source text into a fixed-length vector like standard RNN and LSTM.

A.4.3 Transformer Models

The Transformer was first proposed in the paper "Attention is All You Need" [3]. Transformers are models that uses self-attention to boost the speed with which these models can be trained and are basically an attention mechanism that learns contextual relations between words (or sub-words) in a text. Their advantage comes from how The Transformer lends itself to parallelization, as shown in Figure A.6. A transformer is an encoder-decoder architecture model which uses attention mechanisms to forward a more complete picture of the whole sequence to the decoder at once rather than sequentially as illustrated in the figures below.

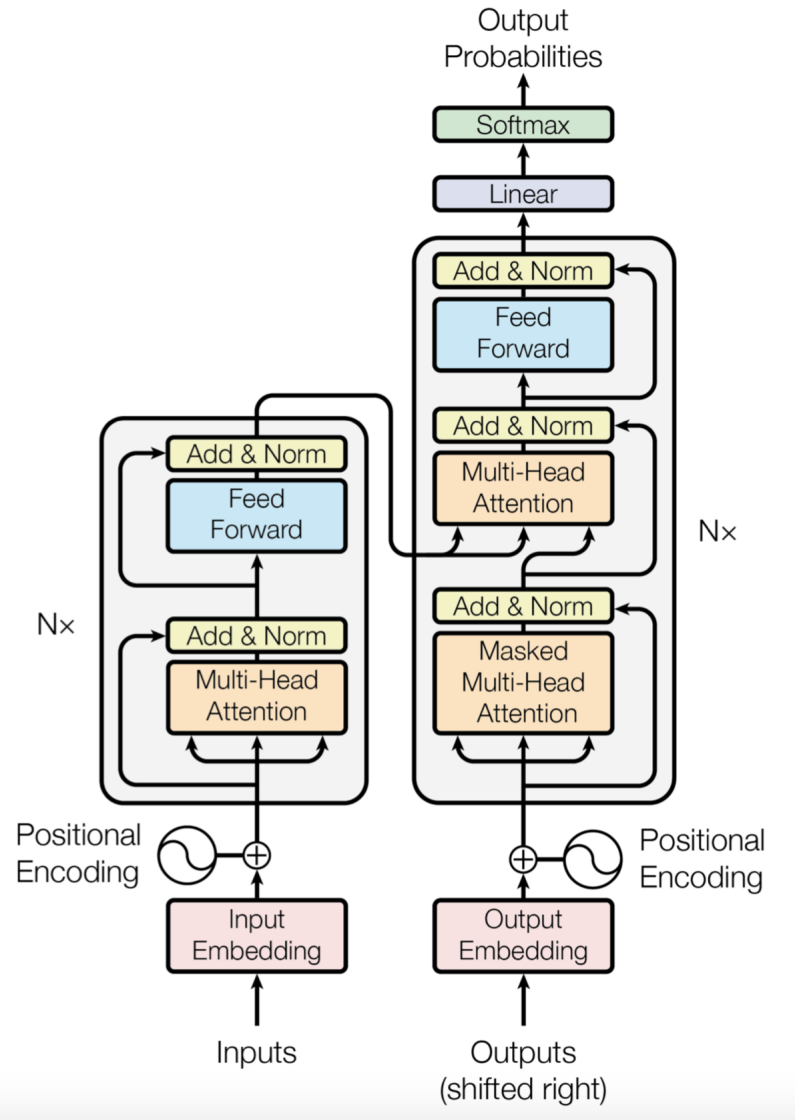


Figure A.6: Transformer Architecture [3].

A.4.4 ELMo

A new type of deep contextualized word representation was presented in [71] that directly addresses challenges mentioned earlier and can be easily integrated into existing models. Elmo significantly improves the state of the art in every considered case across a range of challenging language understanding problems. Elmo generalizes traditional word embedding research along a different dimension. They extract context-sensitive features from a left-to-right and a right-to-left language model. The contextual representation of each token is the concatenation of the left-to-right and right-to-left representations. When integrating contextual word embeddings with existing task-specific architectures, ELMo advanced the state of the art for several major NLP benchmarks.

A.4.5 BERT

BERT stands for Bidirectional Encoder Representations from Transformers. It was designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context [72]. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks. There are many applications that BERT have been applied such as question answering systems, and language inference.

Basically, there are two strategies to apply pre-trained language representations to tasks based on the paper that BERT architecture was introduced [72]: feature-based and fine-tuning. The feature-based approaches such as ELMo [71] uses task-specific architectures that include the pre-trained representations as additional features. Fine-tuning approach, such as the Generative Pre-trained Transformer, introduces minimal task-specific parameters but is trained by fine-tuning all pre-trained parameters. One disadvantage of such models was they have the same objective function during pre-training, where they use unidirectional language models to learn general language representations. This feature limits the choice of architectures that can be used during pre-training. This feature is shown in A.8.

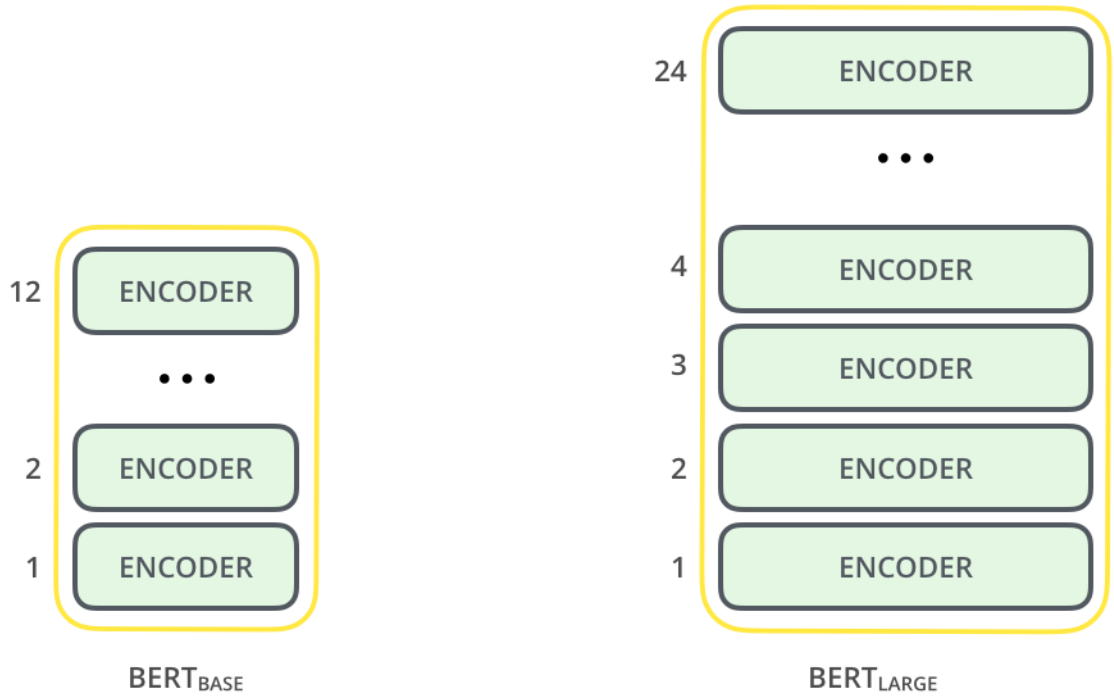


Figure A.7: BERT Models [50]

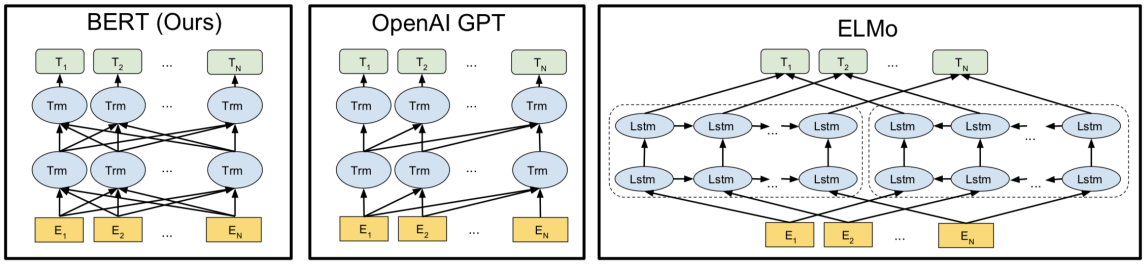


Figure A.8: Differences in pre-training model architectures. BERT uses a bidirectional Transformer [50]. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to- left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers.

A.5 Emotion Models

In the fields of psychology, and sociology, emotions have been studied by prominent researchers such as Ekman and Plutchik [76, 77]. Even though many psychologists have accepted the theory of basic emotions, there is no consensus about the precise number of basic emotions. Robert Plutchik proposed eight primary emotions: anger, fear, sadness, disgust, surprise, anticipation, trust and joy, and arranged them in a color wheel called *Wheel of Emotion* as shown in Figure A.9. Ekman proposed seven basic emotions: fear, anger, joy, sad, contempt, disgust, and surprise; but he changed to six basic emotions: fear, anger, joy, sadness, disgust, and surprise.

In 1980, Robert Plutchik diagrammed a wheel of eight emotions: joy, trust, fear, surprise, sadness, disgust, anger and anticipation, inspired by his Ten Postulates. Plutchik also theorized twenty-four "Primary", "Secondary", and "Tertiary" dyads (feelings composed of two emotions). The wheel of emotions can be paired in four groups:

- Primary dyad = one petal apart = Love = Joy + Trust
- Secondary dyad = two petals apart = Envy = Sadness + Anger
- Tertiary dyad = three petals apart = Shame = Fear + Disgust
- Opposite emotions = four petals apart = Anticipation + Surprise

Opposites: Each primary emotion has a polar opposite, so that:

- Joy is the opposite of sadness.
- Fear is the opposite of anger.
- Anticipation is the opposite of surprise.
- Disgust is the opposite of trust.

Figure A.9: Plutchik's Wheel of Emotions

A.6 Classification

A.6.1 Convolutional Neural Networks (CNNs)

The Convolutional Neural Network (CNN) is a special type of feedforward neural network originally employed in the field of computer vision. Its design is inspired by the human visual cortex, a visual mechanism in animal brain.

Convolutional layers in a CNN play the role of feature extractor, which extracts local features as they restrict the receptive fields of the hidden layers to be local. It means that CNN has a special spatially- local correlation by enforcing a local connectivity pattern between neurons of adjacent layers. Such a characteristic is useful for classification in NLP, in which we expect to find strong local clues regarding class membership, but these clues can appear in different places in the input. For example, in a document classification task, a single key phrase can help in determining the topic of the document. We would like to learn that certain sequences of words are good indicators of the topic, and do not necessarily care where they appear in the document. Convolutional and pooling layers allow the CNN to learn to find such local indicators, regardless of their positions.

A.6.2 Recurrent Neural Network

Recurrent Neural Network (RNN) is a class of neural networks whose connections between neurons form a directed cycle. Unlike feedforward neural networks, RNN can use its internal "memory" to process a sequence of inputs, which makes it popular for processing sequential information. The "memory" means that RNN performs the same task for every element of a sequence with each output being dependent on all previous computations, which is like "remembering" information about what has been processed so far.

Figure A.10 shows an example of a RNN. The left graph is a folded network with cycles, while the right graph is an unfolded sequence network with multiple time steps. The length of time steps is determined by the length of input. For example, if the word sequence to be processed is a sentence of six words, the RNN would be unfolded into a neural network

with six time steps or layers.

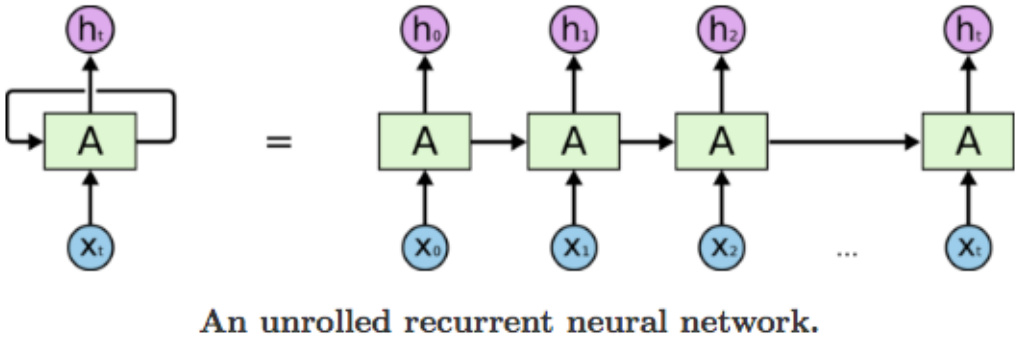


Figure A.10: A simple RNN

The hidden state is regarded as the memory of the network. It captures information about what happened in all previous time steps. Theoretically, RNN can make use of the information in arbitrarily long sequences, but in practice, the standard RNN is limited to looking back only a few steps due to the vanishing gradient or exploding gradient problem.

A.6.3 LSTM

Long Short Term Memory network (LSTM) is a special type of RNN, which is capable of learning long-term dependencies [78]. All RNNs have the form of a chain of repeating modules. In standard RNNs, this repeating module normally has a simple structure. However, the repeating module for LSTM is more complicated. Instead of having a single neural network layer, there are four layers interacting in a special way. Besides, it has two states: hidden state and cell state.

A.6.4 Bidirectional RNN

Researchers have developed more sophisticated types of RNN to deal with the shortcomings of the standard RNN model: Bidirectional RNN, Deep Bidirectional RNN and Long Short Term Memory network. Bidirectional RNN is based on the idea that the output at each

time may not only depend on the previous elements in the sequence, but also depend on the next elements in the sequence. For instance, to predict a missing word in a sequence, we may need to look at both the left and the right context. A bidirectional RNN consists of two RNNs, which are stacked on the top of each other. The one that processes the input in its original order and the one that processes the reversed input sequence. The output is then computed based on the hidden state of both RNNs. Deep bidirectional RNN is similar to bidirectional RNN. The only difference is that it has multiple layers per time step,which provides higher learning capacity but needs a lot of training data.

A.7 Emotion Analysis

Emotions are the subjective feelings and thoughts of human beings. The primary emotions include love, joy, surprise, anger, sadness and fear. The concept of emotion is closely related to sentiment. For example, the strength of a sentiment can be linked to the intensity of certain emotion like joy and anger. Thus, many deep learning models are also applied to emotion analysis following the way in sentiment analysis. In [79], they built a bilingual attention network model for code-switched emotion prediction. A LSTM model is used to construct a document level representation of each post, and the attention mechanism is employed to capture the informative words from both the monolingual and bilingual contexts.

The work in [80] proposed an emotional chatting machine to model the emotion influence in large-scale conversation generation based on GRU. The technique has also been applied in other papers. In [81], the researchers proposed an approach to use millions of emoji occurrences in social media for pretraining neural models in order to learn better representations of emotional contexts. A question-answering approach is proposed using a deep memory network for emotion cause extraction in [82]. Emotion cause extraction aims to identify the reasons behind a certain emotion expressed in text.

Emotion Lexicons

Words can be associated with different intensities (or degrees) of an emotion. For example, most people will agree that the word *condemn* is associated with a greater degree of anger (or more anger) than the word *irritate*. In this work, they created an affect intensity lexicon with real-valued scores of association using best–worst scaling. They refer to this lexicon as the NRC Emotion/Affect Intensity Lexicon.

- The NRC Emotion/ Affect Intensity Lexicon (version 1) is a list of English words with real-valued scores of intensity for eight basic emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, and trust)
- The NRC Word-Emotion Association Lexicon is a list of English words and their associations with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive), the annotations were manually done by crowd sourcing.

Bibliography

Bibliography

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[2] J. Yoon and H. Kim, "Multi-channel lexicon integrated cnn-bilstm models for sentiment analysis," in *Conf on Computational Linguistics and Speech Processing (ROCLING)*, Taipei, Taiwan, 2017.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[4] J. Donath, *The Social Machine*. MIT Press, 2014.

[5] Z. Rajabi, A. Shehu, and A. Purohit, "User behavioral modeling for fake information mitigation on social web," in *SBP-BRiMS*. Springer, 2019, pp. 234–244.

[6] E. M. Bender and A. Koller, "Climbing towards NLU: On meaning, form, and understanding in the age of data," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 5185–5198. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.463

[7] "The next era of human-machine partnerships," in *Emerging Technologies' Impact on Society or Work in 2030*, Institute for the Future, Ed. Dell Technologies, 2017, pp. 1–23.

[8] L. Quijano-Sanchez, J. A. Recio-Garciá, and B. Díaz-Agudo, "HappyMovie: A Facebook application for recommending movies to groups," in *Intl Conf on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2011, pp. 239–244.

[9] N. G. Meshram and A. P. Bhagat, "Connotative features based affective movie recommendation system," in *Intl Conf on Information Communication and Embedded Systems*. IEEE, 2014, pp. 1–7.

[10] A. Gayed, J. S. Milligan-Seville, J. Nicholas, B. T. Bryan, A. D. LaMontagne, A. Milner *et al.*, "Effectiveness of training workplace managers to understand and support the mental health needs of employees: a systematic review and meta-analysis," *Occup Environ Med*, vol. 75, no. 6, pp. 462–470, 2017.

[11] C. Wang and H. Lin, "Constructing an affective tutoring system for designing course learning and evaluation," *J of Educational Computing*, vol. 55, no. 8, pp. 1111–1128, 2017.

[12] F. Catania, M. Spitale, D. Fisicaro, and F. Garzotto, "CORK a conversational agent framework exploiting both rational and emotional intelligence," in *Intelligent User Interfaces (IUI) Workshops*. ACM, 2019, pp. 1–8.

[13] A. S. Zahra Rajabi and O. Uzuner, "A multi-channel bilstm-cnn model for multilabel emotion classification of informal text," 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9031463

[14] H. Purohit and R. Pandey, "Intent mining for the good, bad, and ugly use of social web: Concepts, methods, and challenges," in *Emerging Research Challenges and Opportunities in Computational Social Network Analysis and Mining*. Springer, 2019, pp. 3–18.

[15] H. Allcott and M. Gentzkow, "Social media and fake news in the 2016 election," *J Economic Perspectives*, vol. 3, no. 2, pp. 211–236, 2017.

[16] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," *ACM SIGKDD Explorations Newsletter*, vol. 19, no. 1, pp. 22–36, 2017.

[17] K. M. Carley, G. Cervone, N. Agarwal, and H. Liu, "Social cyber-security," in *SBP-BRiMS*. Springer, 2018, pp. 389–394.

[18] S. Al-khateeb, M. N. Hussain, and N. Agarwal, "Social cyber forensics approach to study twitter's and blogs' influence on propaganda campaigns," in *SBP-BRiMS*. Springer, 2017, pp. 108–113.

[19] K. Bock, S. Shannon, Y. Movahedi, and M. Cukier, "Application of routine activity theory to cyber intrusion location and time," in *2017 13th European Dependable Computing Conference (EDCC)*, 2017, pp. 139–146.

[20] P.-M. Hui, C. Shao, A. Flammini, F. Menczer, and G. L. Ciampaglia, "The hoaxy misinformation and fact-checking diffusion network," in *ICWSM*, 2018.

[21] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Communications of the ACM*, vol. 59, no. 7, pp. 96–104, 2016.

[22] K. Starbird, "Examining the alternative media ecosystem through the production of alternative narratives of mass shooting events on twitter." in *ICWSM*, 2017, pp. 230–239.

[23] M. Viviani and G. Pasi, "Credibility in social media: opinions, news, and health information—a survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 5, p. e1209, 2017.

[24] M. Sameki, T. Zhang, L. Ding, M. Betke, and D. Gurari, "Crowd-o-meter: Predicting if a person is vulnerable to believe political claims," in *HCOMP*, 2017, pp. 157–166.

[25] M. Farajtabar, J. Yang, X. Ye, H. Xu, R. Trivedi, E. Khalil, S. Li, L. Song, and H. Zha, "Fake news mitigation via point process based intervention," in *34th Int'l Conf. on Machine Learning,*. JMLR.org, 2017, pp. 1097–1106.

[26] L. Jin, Y. Chen, T. Wang, P. Hui, and A. V. Vasilakos, "Understanding user behavior in online social networks: A survey," *IEEE Communications Magazine*, vol. 51, no. 9, pp. 144–150, 2013.

[27] M. Jiang, P. Cui, and C. Faloutsos, "Suspicious behavior detection: Current trends and future directions," *IEEE Intelligent Systems*, vol. 31, no. 1, pp. 31–39, 2016.

[28] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini, "Online human-bot interactions: Detection, estimation, and characterization," in *ICWSM*, 2017, pp. 280–289.

[29] M. Heidari, J. H. J. Jones, and O. Uzuner, "Deep contextualized word embedding for text-based online user profiling to detect social bots on twitter," in *IEEE 2020 International Conference on Data Mining Workshops (ICDMW), ICDMW 2020*, 2020.

[30] M. Heidari and J. H. J. Jones, "Using bert to extract topic-independent sentiment features for social media bot detection," in *IEEE 2020 11th Annual Ubiquitous Computing, Electronics Mobile Communication Conference, UEMCON 2020*, 2020.

[31] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, E. M. Joo, D. Weiping, and L. Chin-Teng, "A review of clustering techniques and developments," *Neurocomputing*, vol. 267, pp. 664–681, 2017.

[32] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*. Association for Computational Linguistics, 2002.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[34] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *J Artificial Intel Res*, vol. 16, pp. 321–357, 2002.

[35] Z. Rajabi, A. Shehu, and H. Purohit, "User behavior modelling for fake information mitigation on social web," in *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*. Springer, 2019, pp. 234–244.

[36] M. Bouazizi and T. Ohtsuki, "A pattern-based approach for multi-class sentiment analysis in Twitter," *IEEE Access*, vol. 5, pp. 20 617–20 639, 2017.

[37] S. M. Mohammad, F. Bravo-Marquez, M. Salameh, and S. Kiritchenko, "Semeval-2018 Task 1: Affect in tweets," in *International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA, 2018.

[38] P. Ekman, "An argument for basic emotions," *Cognition and Emotion*, vol. 6, no. 3-4, pp. 169–200, 1992.

[39] R. Plutchik, *Emotion: A Psychoevolutionary Synthesis*. Harper & Row, 1980.

[40] J. A. Russell, "A circumplex model of affect," *J Personality & Social Psychology*, vol. 39, pp. 1161–1178, 1980.

[41] L. Bostan and R. Klinger, "An analysis of annotated corpora for emotion classification in text," in *Intl Conf on Computational Linguistics*, Santa Fe, Mexico, 2014.

[42] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119.

[43] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representations," in *Conf on Empiral Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014.

[44] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[45] K. Yoon, "Convolutional neural networks for sentence classification," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[46] F. A. Acheampong, C. Wenyu, and H. Nunoo-Mensah, "Text-based emotion detection: Advances, challenges, and opportunities," *Engineering Reports*, vol. 2, no. 8, p. e12189, 2020.

[47] F. A. Acheampong, H. Nunoo-Mensah, and C. Wenyu, "Transformer models for text-based emotion detection: a review of BERT-based approaches," *Artificial Intelligence Review*, 2021.

[48] D. Demszky, D. Movshovitz-Attias, J. Ko, A. Cowen, G. Nemade, and S. Ravi, "Goemotions: A dataset of fine-grained emotions," 2020.

[49] W. Gao and F. Sebastiani, "Tweet sentiment: From classification to quantification," in *IEEE/ACM Intl Conf Adv Social Netw Anal Mining (ASONAM)*, 2015, pp. 97–104.

[50] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, J. Burstein, C. Doran, and T. Solorio, Eds. Assoc Comput Linguistics, 2019, pp. 4171–4186.

[51] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018.

[52] Y. Huang, S. Lee, M. Ma, Y. Chen, Y. Yu, and Y. Chen, "Emotionx-idea: Emotion BERT - an affectional model for conversation," *arXiv preprint arXiv:1908.06264*, 2019.

[53] A. Radford, "Improving language understanding by generative pre-training," 2018.

[54] "Class-balanced loss based on effective number of samples," in *IEEE/CVF Conf on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9268–9277.

[55] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *arXiv preprint arXiv:1708.02002*, 2017.

[56] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv preprint arXiv:1807.05118*, 2018.

[57] J. M. D Jurafsky, *Speech and language processing (3rd edition)*. Prentice Hall, 2019.

[58] E. W. Noreen, in *Computer Intensive Methods for Testing Hypothesis*. Wiley, 1989.

[59] B. Efron and R. J. Tibshirani, in *An introduction to the bootstrap*. CRC press, 1993.

[60] F. eight, "Data for everyone," 2019. [Online]. Available: https://www.figure-eight.com/data-for-everyone/

[61] Crowdflower, "Sentiment analysis in text," 2019. [Online]. Available: https://data.world/crowdflower/sentiment-analysis-in-text

[62] L.-A.-M. Bostan and R. Klinger, "An analysis of annotated corpora for emotion classification in text," in *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 2104–2119. [Online]. Available: https://www.aclweb.org/anthology/C18-1179

[63] M. Bouazizi and T. Ohtsuki, "Sentiment analysis: From binary to multi-class classification: A pattern-based approach for multi-class sentiment analysis in Twitter," in *IEEE Intl Conf on Communications*, 2016, pp. 1–6.

[64] T. G. Dietterich, "Ensemble methods in machine learning," in *First Intl Workship on Multiple Classifier Systems (MCS)*, 2000.

[65] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable ai: A review of machine learning interpretability methods," *Entropy*, vol. 23, no. 18, pp. 1–45, 2002.

[66] M. Danilevsky, K. Qian, R. Aharonov, Y. Katsis, B. Kawas, and P. Sen, "A survey of the state of explainable ai for natural language processing," in *1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*. Suzhou, China: Association for Computational Linguistics, December 2020, pp. 447–459.

[67] S. Serrano and N. A. Smith, "Is attention interpretable?" in *57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 2931—-2951.

[68] S. Mohammad, "Word affect intensities," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[69] M. Porter, "An algorithm for suffix stripping," vol. 14, no. 3, p. 130137, 1980.

[70] M. Joos, "Description of language design," in *Journal of Systems Architecture*, no. 3. JSA, 1950, p. 701–708.

[71] 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. [Online]. Available: http://dx.doi.org/10.18653/v1/N18-1202

[72] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.

[73] 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)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://www.aclweb.org/anthology/D14-1162

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

[75] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3104–3112. [Online]. Available: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[76] P. Ekman, "An argument for basic emotions," *Cognition and Emotion*, vol. 6, no. 3-4, pp. 169–200, 1992.

[77] R. Plutchik, *Emotion: A Psychoevolutionary Synthesis*. Harper & Row, 1980.

[78] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[79] Z. Wang, Y. Zhang, S. Lee, S. Li, and G. Zhou, "A bilingual attention network for code-switched emotion prediction," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 1624–1634. [Online]. Available: https://www.aclweb.org/anthology/C16-1153

[80] H. Zhou, M. Huang, T. Zhang, X. Zhu, and B. Liu, "Emotional chatting machine: Emotional conversation generation with internal and external memory," 2018.

[81] 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," *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017. [Online]. Available: http://dx.doi.org/10.18653/v1/D17-1169

[82] L. Gui, J. Hu, Y. He, R. Xu, L. Qin, and J. Du, "A question answering approach for emotion cause extraction," *Proceedings of the 2017 Conference on*

Empirical Methods in Natural Language Processing, 2017. [Online]. Available: http://dx.doi.org/10.18653/v1/D17-1167

Curriculum Vitae

B.1 Curriculum Vitae

zrajabi@gmu.edu **www.linkedin.com/in/Zahra-Rajabi**

A passionate machine learning researcher and PhD candidate with 10+ years of experience in **Deep Learning, NLP and Machine Learning** and demonstrated expertise in the problems such as Image Retrieval and Search, Natural Language Processing and Social Computing. I had a valuable internship experience at **RedAdobe** company as an *Applied Machine Learning Engineer* during Summer and Fall 2020. I will graduate in May 2021 and I am actively looking for a position as an **Applied/ Research Scientist**.

B.1.1 TECHNICAL SKILLS

- ***Deep learning Frameworks:*** TensorFlow, Keras, PyTorch, Scikit-learn, Caffe
- ***AWS services:*** AWS EC2, Amazon S3, CUDA
- ***Programming Languages:*** Python, Java, MATLAB, C++
- ***Machine Learning/Computer vision****:* OpenCV, MATConvNet, sckitlearn, vlFeat.
- ***Graph Visualization and Network Analysis:*** igraph, NetworkX, Gephi
- ***Optimization and Integer Programming:*** GurobiPy
- ***Natural Language Processing (NLP) libraries/Language Models:*** FastText, NLTK, Gensim/ BERT,

B.1.2 EDUCATION

George Mason University (GMU) GPA: 3.63
PhD in Information Technology (Research in NLP, AI and Deep Learning). 2014-2021
MSc. in Computer Science (AI) GPA: 3.6 2014-2018

Amirkabir University of Technology (AUT), Tehran, Iran.
MSc. in Information Technology Fall 2009

Shahid Beheshti University, Tehran, Iran.
B.S. in Software Engineering. Fall 2006

B.1.3 RELEVANT WORK EXPERIENCE

Applied Machine Learning Engineer - Internship*, *Adobe May 2020- Jan 2021
I have been working to build a **Query Dependent Ranker** model for **Adobe Stock Search** service. We used triplet ranking loss and combined Google's Inception-V3 Architecture with different language models for visual semantic representation learning. Our results has shown improvement in ranking text query image retrieval.

Applied Machine Learning Researcher-Summer Internship, *Text Research Team at* ***Digital Reasoning*** *Company* June- August 2019

- Developed **Deep Learning** models for **Insider Threat Detection** project and Conduct Surveillance at DR company. Different embeddings were explored such as FastText, GloVe, Word2Wec and Doc2Vec and I developed several language models to classify users based on their Myers Briggs Type Indicator Personality Test (MBTI) scores on PersonalityCafe Forum.
- Multiclass **Semantic Analysis** and **Multi-label Emotion Classification**: Inspired by kim-CNN for sentence classification, I developed a **Multi-Channel Multi-filter BiLSTM-CNN** model for Multi-label ***Twitter*** Emotion Classification using SemEval2018 Affect in Tweets dataset and wiki-GloVe word embeddings.

Graduate Research Assistant, *Humanitarian, Semantics and Info. Lab.* 2018-June 2019

- **User Behavior Modelling for Fake Information Mitigation on Twitter**
  - Developed an **unsupervised learning** setup to identify key user behavior categories and categorized user types using user, network, and content features.
  - Built predictive models for user behavior estimation using RBF SVM, Neural Networks, etc. and applying **Deep Learning LSTM** architecture for sequence prediction to capture temporal patterns of user activities.
- **Community Detection for Decoy Selection in Template-free Protein Structure Prediction**: we leveraged different community detection methods on graph-based embedding of decoys to organize protein structure spaces probed in silico. We comprehensively evaluated them and showed Louvain and Greedy Modularity Maximization methods outperform other community detection methods on decoy selection. 2017-2018
- **Persistent Service for a Swarm of UAVs:** NSF Smart and Autonomous Systems.

  - Developed a multi-objective motion planning algorithm for a swarm of UAVs for covering task an Emergency Responder system.
  - Developed a **Multi-threaded simulator for real-time Queuing and Scheduling of multiple UAVs** [Python, GurobiPy] 2016-2017

- **Confidence Estimation in Stem Cell Classification**, National Institute of Standards and Technology **(NIST)** 2014-2015

Graduate Teaching Assistant Department of Computer Science 2015-2018
Courses: Data Structures, Computer Vision, Advanced Computer Vision, Java Programming Duties: Grading projects, writing JUnit tests, holding weekly office hours, administering labs.

B.1.4 SELECTED PUBLICATIONS

- Zahra Rajabi, Ozlem Uzuner, Amarda Shehu, ***Meta-Learning to Extract Emotions from Short Text.*** Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021), August 1-6, 2021. Bangkok, Thailand. *In preparation; to be submitted by April 26.*

- Zahra Rajabi, Ozlem Uzuner, Amarda Shehu, ***Detecting Scarce Emotions Using BERT and Hyperparameter Optimization***. The 30th International Conference on Artificial Neural Networks, September 14-17, 2021. Virtual Event. *Submitted.*

- Zahra Rajabi, Ozlem Uzuner, Amarda Shehu, ***Beyond Binary Sentiments: A Multi-Channel BiLSTM-CNN model for Multilabel Emotion Classification of Informal Text***. 14th IEEE International Conference on Semantic Computing, February 3-5, 2020. San Diego, California, USA.

- Zahra Rajabi, Amarda Shehu, Hemant Purohit, ***User Behavioral Modeling for Fake Information Mitigation on Social Web.*** 2019 International Conference on Social Computing, Behavioral-Cultural Modeling, & Prediction and Behavior Representation in Modeling and Simulation, SBP-BRiMS 2019, George Washington University, Washington DC, USA.

- Kazi Lutful Kabir, Liban Hassan, Zahra Rajabi, Nasrin Akhter and Amarda Shehu ***Graph-Based Community Detection for Decoy Selection in Template-Free Protein Structure Prediction***, Journal of Molecules 24 (5), 854.

- Liban Hassan, Zahra Rajabi, Nasrin Akhter, and Amarda Shehu. ***Community Detection for Decoy Selection in Template-free Protein Structure Prediction***. The 11th Computational Structural Bioinformatics Workshop (CSBW-2018), Washington, D.C. 2018.

- Nasrin Akhter, Liban Hassan, Zahra Rajabi, Daniel Barbara, and Amarda Shehu. ***Learning Organizations of Protein Energy Landscapes: An Application on***

Decoy Selection in Template-Free Protein Structure Prediction. In Methods in Molecular Biology: Protein Supersecondary Structure (Springer), first edition, (Editor: Kister, A.), 2018.

- Rajabi Z, Kosecka J, Bajcsy P: ***Confidence Estimation in Stem Cell Classification***. BioImage Informatics Conference 2015, National Institute of Standards and Technology (NIST), Gaithersburg, MD.
- Zahra Rajabi, Ali akbar Alamdari, Abdollah Doosti Aref and Rabe Karimi Mahabadi, ***"Special Topics in Electrical and Computer Engineering with MATLAB: Neural Network, Image Processing, Fuzzy Logic, Genetic Algorithms"***, published by Negarande Danesh publication, ISBN 978-600-6190-10-5, 2011.
- Z. Rajabi, S. Zareh, M. Shamsfard, **"Logic-Based Persian Language Sentences Representation and Generation System"**, the 12th International CSI Computer Conference (CSICC'07), Shahid Beheshti University, Tehran, Iran.

B.1.5 SELECTED GRADUATE COURSEWORK

Computer Science: Data Mining, Machine Learning, Network Science, Analysis of Algorithms, Autonomous Robotics, Pattern Recognition, Planning Motions of Robots, Computer Vision.
Operation Research:Integer Programming, Stochastic Models.

B.1.6 OLDER RESEARCH EXPERIENCE

Computer Vision

1. Object Detection with Supervised Object Proposals. (Dataset: Washington RGB-D Scenes v2 dataset)
2. Object Discovery in Indoor Scenes on RGB-D images (Dataset: WRGB-D)
3. Object Recognition using SURF features vs. Deep Learning Dataset: ImageNet
4. **Deep Face Recognition** using **MatConvNet (VGG-Face CNN descriptor)**
5. **Face Tracking** using **Kanade-Lcucas-Tomasi (KLT)** algorithm (MATLAB)
6. **Face Recognition** using BLP features **Python and OpenCV**
7. Nonparametric Probability Density Estimation and Image Segmentation using Expectation Maximization (MATLAB)

Data mining

1. Discovery of Patterns of Cascading Behaviors On Twitter (Dataset: tweets of 2016 US president election). Interaction Network, Infection propagation of tags was modeled using **C++, igraph, gephi visualization**.

2. Community Detection Algorithms based on stochastic modularity maximization, spectral clustering and random walks. (Python, Datasets: **ego-Facebook**)

3. **Multi-Objective Canadian Armed Force Manpower Optimization Model** and development of the staffing plan using **Goal Programming** using (Python, Gurobi)

B.1.7 EXTRA-CURRICULAR ACTIVITIES

- **ACM BCB Student Volunteer, Washington, D.C.** 2018

- **https://www.unicefusa.org/stories/introducing-unicef-congressional-action-teams/26971UNICEF DC Chapter, Congressional Action Team (ACT)**, June 2016-Present
  Volunteer member to advocate and support world's children and help engage legislators to support this cause regarding legislation.

- **IranWiC (Iranian Women in Computing) Committees Volunteer** George Mason Representative 2019-Present

- **Iranian Student Union (ISU)** 2016-2017
  Treasurer, George Mason University, Fairfax, VA.

B.1.8 HONORS

- **GHC Scholarship Winner** Grace Hopper Celebration. Orlando, Florida, October 2019.

- **CRA-W Grad Cohort Scholarship Winner** San Francisco 2015 and 2018, Computing Research Association - Women.

- Ranked as top % 0.2 in BSc Nationwide Universities Entrance Exam in the field of Mathematics & Physics, Iran. (1022th among more than 500,000).

- Ranked as top % 0.03 in MSc Nationwide Universities' Entrance Exam, Iran. (34th among more than 100,000), and got admission from Amirkabir University of Technology, the best university in Information Technology.