Human Emotion and Sentiment in Natural Language Understanding and Generation using Large Language Models with Limited to No Labeled Data

by

Md Riyadh

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Information Technology

Carleton University

Ottawa, Ontario

Acknowledgments

I would like to express my heartfelt gratitude to everyone who has supported me throughout my doctoral journey.

First and foremost, I am deeply grateful to my supervisor, Professor Omair Shafiq, for his guidance, expertise, and unwavering support. His insightful and constructive feedback have been invaluable in shaping my research and helping me grow as a researcher. With his excellent mentorship, I was able to publish several conference and journal papers in reputed venues. In addition to academic guidance, he has been exceptionally supportive in helping me successfully navigate many unique challenges I encountered, including the tough time shared by all of us during the COVID-19 pandemic. Without his phenomenal support, I would not have been able to complete this endeavor.

I would like to thank the members of my thesis committee for their time and effort in reviewing my work and providing valuable feedback. I would also like to thank Professor Ali Arya for his kind support during the initial phase of my Ph.D., without which I would not have embarked on this journey.

My sincere thanks also go to my friends who have offered encouragement, assistance, and camaraderie throughout this challenging but rewarding journey. Particularly, I would like to thank my dear friend Dr. Gerry Chan for his consistent encouragement and support throughout my Ph.D. journey. His inspiration has been a key factor behind my decision to undertake this challenge and then persist in it. I would also like to thank my colleagues and mentors who have supported me in various capacities during this journey.

Special thanks are due to my family, who have been my pillars of strength and unwavering support through thick and thin. Their love, patience, and understanding have made all the difference in helping me pursue my academic goals. Specifically, I am grateful to my parents for their continuous prayers and inspiration in all my academic pursuits, including this one. It is them who nurtured my curiosity at an early age and turned me into a lifelong seeker of knowledge. I am thankful to my wife who has made remarkable sacrifices and supported me to stay motivated during the entire journey. I am also grateful to my son who is just about to turn 7 and has seen me as a graduate student almost all his life. Though imperceptible, the sacrifices he made by skipping

that play session or that outing is monumental, a lot of which are alleviated by his mother's extraordinary sacrifice and affection.

Thank you all for your support, encouragement, and inspiration. I am deeply grateful and humbled by your kindness and generosity.

Author Declaration

I am the first author of the publications that contribute to this thesis. I used the word 'we' in this document to indicate the collaboration with my supervisor and mentorship from him in completing this research. I want to thank my supervisor, Professor Omair Shafiq, for his support in accomplishing this work. Publications directly related to this thesis include:

- Riyadh, M., & Shafiq, M. O. (2021). Towards Multi-class Sentiment Analysis with Limited Labeled Data. In 2021 IEEE International Conference on Big Data (Big Data) (pp. 4955-4964)
- Riyadh, M., & Shafiq, M. O. (2022). GAN-BElectra: Enhanced Multi-class Sentiment Analysis with Limited Labeled Data. Taylor and Francis Journal of Applied Artificial Intelligence, 36(1), 2083794, impact factor: 2.777.
- Riyadh, M., & Shafiq, M. O. (2022). Towards Emotion Cause Generation in Natural Language Processing using Deep Learning. In 21st IEEE International Conference on Machine Learning and Applications (ICMLA), 2022.
- **[In review]** Riyadh, M., & Shafiq, M. O. (2023). ECSGen and iZen: A New NLP Task and A Zero-shot Framework to Perform It.
- **[Accepted]** Riyadh, M., & Shafiq, M. O. (2023). Towards Automatic Evaluation of NLG Tasks using Conversational Large Language Models. In 19th IFIP International Conference on Artificial Intelligence Applications and Innovations.

Abstract

Natural Language Processing (NLP) aims to utilize computational resources to comprehend and generate human language. Emotion and sentiment are integral parts of human beings, and they are often reflected in human language. Consequently, these two closely related ideas are of paramount importance to NLP. In this thesis, we focus on several NLP tasks related to human emotion and sentiment. Particularly, we focus on the domains of Sentiment Analysis and Emotion-Cause Analysis (ECA). Like most other NLP tasks, machine learning technologies are frequently leveraged to perform various NLP tasks in these two domains. A common challenge in applying machine learning technology to context-dependent tasks like Sentiment Analysis is that they require a large amount of labeled data to develop a performant model. In this thesis, we develop several techniques leveraging Transformer-based large language models (LLMs) to perform various NLP tasks within these two domains in a limited to no labeled data setting. Specifically, we devise two technical architectures to perform multi-class Sentiment Analysis with limited labeled data. We introduce two new NLP tasks within the domain of ECA, which are also the first Natural Language Generation (NLG) tasks in this domain. We devise technical solutions to perform these NLG tasks, one with limited labeled data, and the other with no labeled data. We publish a new dataset for one of these novel NLG tasks. Lastly, we propose leveraging conversational LLMs for the automatic evaluation of open-ended NLG tasks, which also does not require any new training or labeled data.

Keywords: Natural Language Processing, Natural Language Generation, Sentiment Analysis, Emotion-Cause Analysis, Deep Learning, Transformer, Large Language Model

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Meaning
AI	Artificial Intelligence
ANN	Artificial Neural Networks
BERT	Bidirectional Encoder Representations from Transformers
BOW	Bag of Words
CBOW	Continuous Bag of Words
CNN	Convolution Neural Networks
CPU	Central Processing Unit
CRF	Conditional Random Field
DNN	Deep Neural Network
ECE	Emotion cause extraction
ECPE	Emotion-Cause Pair Extraction
ESCP	Emotion-Cause Span-Pair extraction and classification
GAN	Generative Adversarial Network
GPT	Generative Pretrained Transformer
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
IDF	Inverse Document Frequency
LLM	Large Language Model
LSTM	Long Short-Term Memory
NB	Naive Bayes
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
OPT	Open Pretrained Transformer
POS	Parts of Speech
RF	Random Forest
RNN	Recurrent Neural Networks
SGD	Stochastic Gradient Descent
SOTA	State of the Art
SRN	Simple Recurrent Network
SS	Semi Supervised
SVM	Support Vector Machine
TF	Term Frequency
TPU	Tensor Processing Unit

Chapter 1: Introduction

In this chapter, we provide an overview of this thesis, including the main research problems we investigated along with our associated contributions. We discuss the common premises and distinctiveness of these contributions. We also outline the overall structure of this document.

1.1 Overview

With the internet and social media rapidly evolving in recent years, a vast amount of text data is inundating every aspect of our lives. Much of this textual data is unstructured and raw, making it difficult to properly utilize. One of the most significant challenges in Artificial Intelligence (AI) is developing systems that can comprehend unstructured text data and reasonably deduce various rich information from them. This is an important driving factor for the rapid advancement of the Natural Language Processing (NLP) research field that we currently witness. NLP is an interdisciplinary subfield of linguistics and computer science, which involves understanding and generation of human language using AI. Natural Language Understanding (NLU) [1], a subset of NLP, concentrates on assessing and comprehending human language, including named entities, sentiment, and intent. On the other hand, the generation of natural language, like summaries, inquiries, or descriptions of an entity, falls under the category of Natural Language Generation (NLG) from [1], which is another subfield of NLP. Sophisticated machine learning systems are being developed and enhanced every day at an unprecedented pace to perform various NLP tasks such as Sentiment Analysis, machine transition, question-answering, summary generation, and emotion analysis among other applications.

In this thesis, we refer to NLP as it relates to human language in the textual form. One of the underlying themes of the research problems we investigate in this thesis is the application domains we explore, which revolve around human sentiment and emotion. Another common premise is the quest of performing specific NLP tasks with limited to no training data. All the technical solutions we propose to perform various NLP tasks in this thesis intersect at the involvement of Transformer-based Large Language Models (LLMs). We explore both NLU and NLG as part of our investigation. Specifically, for NLU, we investigate multi-class Sentiment Analysis with limited label data, and for NLG, we explore the domain of Emotion-Cause Analysis (ECA). While exploring these different avenues of NLP with an emphasis on the underlying themes mentioned

above, we develop new techniques to perform an existing task (i.e., Sentiment Analysis), we propose novel NLP tasks, particularly two new generative NLP tasks (i.e., NLG tasks) within the ECA domain, we curate our own dataset and devise technical solutions to perform these novel tasks. Additionally, we suggest a novel method for the automatic evaluation of open-ended NLG tasks, such as the ones we propose in this thesis.

1.2 Research Problems

In this thesis, we aim to address several research problems which encompass some common premises as well as some distinct elements. The common grounds include the basis of the application domains we explore which are human emotion and sentiment. It also includes our consistent focus on solving NLP tasks with limited to no labeled data with techniques involving Transformer-based large language models (LLMs). With these common premises, each constituent research work of this thesis builds upon the learning and inspiration from the previous one and contributes distinctively to the respective areas. Our thesis commences with Sentiment Analysis which subsequently extends toward analyzing the cause of the emotion. Further investigation into the ECA research area results in the proposal of two novel NLG tasks and the development of associated technical solutions to perform them. Lastly, inspired by the limitations of the existing automatic evaluators for open-ended NLG tasks that we observe while evaluating these newly proposed NLG tasks, we suggest a new automatic evaluation technique for such NLG tasks. We outline these successive research problems in the following paragraphs:

In Sentiment Analysis, a given piece of text is categorized with respect to the sentiment it expresses, often in terms of positive, neutral, or negative. Training machine learning models to perform such classification of text is a common phenomenon. This type of machine learning technique, commonly referred to as supervised learning, heavily relies on training data. Complexity in Sentiment Analysis can increase due to the fact that words used in one context to express certain sentiments may exert different sentiments in different contexts. As a result, training a machine learning model for Sentiment Analysis tasks typically requires training with domain-specific labeled data. Having a large quantity of labeled training data for specific domains can be tedious and expensive. This problem is one of the focus areas of this thesis. Specifically, we attempt to tackle the following research question:

- **RQ1**: How to develop techniques that can perform multi-class Sentiment Analysis with limited training data (i.e., 50 datapoints per class) while still achieving superior classification accuracy compared to the state-of-the-art (SOTA) baseline technique?

In the ECA domain, we observe that researchers have recently proposed several new tasks, all of which can be broadly labeled as classification tasks. Though we have recently witnessed the rapid growth of NLG tasks (i.e., generative NLP tasks) within the broader domain of NLP, such as poetry generation, machine translation, and question-answering, the application of generative machine learning has largely remained unexplored in the ECA domain. We, therefore, investigate this underexplored area; specifically, we attempt to answer the following research questions:

- **RQ2(a)**: Can we formulate a viable novel NLG task within the domain of ECA that aims to generate a meaningful cause of an emotion expressed in a given text?
- **RQ2(b)**: How to perform the novel task mentioned in RQ2(a) by building upon existing machine learning techniques and leveraging the existing ECA datasets?
- **RQ3(a)**: Can we construct a new and feasible NLG task within the ECA domain that aims to generate relevant suggestions to mitigate the cause of a negative emotion stated in a given text?
- **RQ3(b)**: How to devise a technical solution to perform the novel task mentioned in RQ3(a) without requiring any new training or fine-tuning steps?

Evaluating a machine learning model on its ability to perform the NLG tasks we refer to in the above research questions typically requires human participants. This is equally applicable to almost all open-ended NLG tasks such as summary generation, poetry generation, and paraphrasing. This usually impacts the velocity of these research studies as conducting human evaluation can be time-consuming. While there are many attempts to develop automatic evaluation techniques for NLG tasks, none of them have been regarded as a useful alternative to human evaluation. Largely motivated by the challenges we encountered in evaluating the NLG tasks mentioned in RQ2 and RQ3, we conduct a brief investigation into this area and attempt to answer the following research question:

- **RQ4**: Can we leverage Conversational Large Language Models as an automatic evaluator for open-ended NLG tasks?

1.3 Contributions

The overall contributions of this thesis are listed below:

- We propose two successive techniques for multi-class Sentiment Analysis named SG-Elect and GAN-BElectra. Our experiments demonstrate that SG-Elect achieves significantly higher performance (i.e., F1 macro, and F1 weighted average) in multi-class Sentiment Analysis compared to its baseline. Building upon SG-Elect, we develop GAN-BElectra, which significantly reduces the architecture complexity and required training steps compared to SG-Elect, without having any adverse effect in the classification accuracy in a similar limited labeled data setting. This is related to RQ1. We publish one conference paper and one journal paper based on this work:
    - Riyadh, M., & Shafiq, M. O. (2021). Towards Multi-class Sentiment Analysis with Limited Labeled Data. In 2021 IEEE International Conference on Big Data (Big Data) (pp. 4955-4964)
    - Riyadh, M., & Shafiq, M. O. (2022). GAN-BElectra: Enhanced Multi-class Sentiment Analysis with Limited Labeled Data. Taylor and Francis Journal of Applied Artificial Intelligence, 36(1), 2083794, impact factor: 2.777.
- We propose a new generative NLP task within the domain of ECA named Emotion-Cause Generation (ECG). We enhance an existing infilling architecture to perform this task and establish the viability of the proposed task through a technical demonstration. This is related to RQ2(a) and RQ2(b). We published one conference paper related to this contribution:
    - Riyadh, M., & Shafiq, M. O. (2022). Towards Emotion Cause Generation in Natural Language Processing using Deep Learning. In 21st IEEE International Conference on Machine Learning and Applications (ICMLA), 2022.
- We propose Emotion-Cause mitigating Suggestion Generation (ECSGen), a novel generative NLP task within the domain of ECA. We curate and publish a dataset to perform this new task and for potential future research in this area. This is related to RQ3(a). We also propose iZen, a zero-shot framework to perform the ECSGen task. We demonstrate its promising ability to perform this task without any training or fine-tuning through experiments and evaluations. This is related to RQ3(b). This contribution is reported in the following journal paper:

    - [In Review] Riyadh, M., & Shafiq, M. O. (2023). ECSGen and iZen: A New NLP Task and A Zero-shot Framework to Perform It.
- We hypothesize that the conversational Large Language Models can be used as automatic evaluators for open-ended NLG tasks such as the ones we introduce in this thesis. We perform several experiments to investigate our hypothesis. This contribution is related to RQ4 and is included in the following conference paper:
    - [Accepted] Riyadh, M., & Shafiq, M. O. (2023). Towards Automatic Evaluation of NLG Tasks using Conversational Large Language Models. In 19th IFIP International Conference on Artificial Intelligence Applications and Innovations.

1.4 Overall Document Structure

This thesis is structured as a series of research investigations intended to answer the research questions outlined in section 1.2. The organization of this thesis follows an integrated article format [2] consisting of two introductory chapters (Introduction and Background) and a concluding one along with several self-contained chapters which are based on our own published journals, conference papers, or papers awaiting publication. In addition to the experiment, evaluation, and discussion sections, these self-contained chapters introduce each of the research questions in more detail supported by the discussion on the specific related studies. We describe the overall structure of this thesis below:

- **Chapter 1** introduces the overall research problems we investigate in this thesis along with their common premises and distinctiveness.
- **Chapter 2** provides an overview of the technology and application domains explored in this thesis. We discuss two main application domains of this thesis: Sentiment Analysis, and Emotion-Cause Analysis. We present a literature review to show the overall research trends in these two domains. We also introduce the machine learning technologies that we leverage throughout different parts of this thesis.
- **Chapter 3** presents our study on multi-class Sentiment Analysis with limited labeled data and the two successive techniques that we devise to perform this task. We discuss both techniques in tandem to account for the overlaps between them and to demonstrate how the latter technique builds upon the initial one.

- **Chapter 4** focuses on our first investigation related to the ECA domain where we introduce the first-ever NLG task in this domain named Emotion-Cause Generation (ECG) and illustrate the viability of this new task through a technical demonstration.
- **Chapter 5** delves into the details of our subsequent investigation related to the ECA domain where we introduce a second novel NLG task named Emotion-Cause mitigating Suggestion Generation (ECSGen). We also discuss iZen, a zero-shot technical framework that we develop to perform this novel task. Additionally, this chapter introduces the dataset we curated for this study.
- **Chapter 6** presents our proposal and associated experiments to leverage Conversational Large Language Models (LLMs) to evaluate open-ended NLG tasks. In addition to some common NLG tasks, we also evaluate the two novel ECA-related NLG tasks that we introduce in this thesis (ECG, ECSGen) using this method.
- **Chapter 7** summarizes the key findings of this thesis, their implications in the respective domains along with some potential future work.

Chapter 2: Background

Natural Language Processing (NLP) is a branch of Artificial Intelligence (AI) concerned with the utilization of computational resources to understand and generate human language. In this thesis, we refer to NLP as it relates to human language in the textual form. In broader terms, NLP is an interdisciplinary subfield of linguistics, and computer science, which entails handling, comprehending, and generating human language. This thesis focuses on some distinct NLP tasks related to Sentiment Analysis and Emotion-Cause Analysis, which we discuss in the following sections.

In this chapter, we introduce Sentiment Analysis and the Emotion-Cause Analysis domain and summarize the general research trend in these domains. The subsequent chapters present more focused literature reviews specific to the individual research questions. In addition, this chapter also introduces the machine learning technologies that are relevant to this thesis. The literature review presented in this chapter and subsequent chapters is the culmination of our thorough search of references throughout the successive investigations we report in this thesis. Our literature search strategy included snowballing method [3], both forward and backward. We leveraged several tools for our search including Google Scholar [4], and Carleton University Omni Library [5]. We eventually narrowed down our focus to the specific gaps we have identified in the literature which offered us insight to formulate our research questions and the associated contributions.

2.1 Emotion and Sentiment

There have been many attempts in the literature to formulate a general definition of emotion. For example, according to Scherer [6], emotions refer to coordinated and interconnected changes in various bodily systems in response to the evaluation of an internal or external stimulus that is relevant to the organism's primary concerns. Scherer highlights several key features of emotions, including their event focus, appraisal-driven nature, response synchronization, rapidity of change, behavioral impact, intensity, and duration. Emotions are typically elicited by a stimulus event that has significance to the individual. Such appraisal of significance can be intrinsic or extrinsic and is often based on the individual's goals, desires, or needs. Emotions also involve a coordinated response across various bodily systems that correspond to the individual's appraisal of the event. Emotional states are usually intense and have a significant impact on an individual's behavior and

social interactions. They also undergo constant modification and readjustment in response to changing circumstances or evaluations. However, emotions typically have a relatively short duration to avoid taxing the individual's resources and maintain behavioral flexibility.

Fillmore et al. developed a linguistic resource named FrameNet [7], where they suggest that an individual who experiences emotions has a specific emotional state, which may be described in terms of a particular stimulus that elicits it or a category that classifies the type of stimulus. Additionally, there may be circumstances that influence the emotional response or reasons why the stimulus provokes a particular response in the experiencer. Ortony et al. [8] offer another perspective on emotions, defining them as “valenced” (positive-negative) reactions to events, agents, or objects, with their specific nature being determined by the construction of the eliciting situation. Using this theory as a basis, Ghazi et al. [9] put forth a few propositions to explain the distinctions between some basic emotions. For example, they define “joy” as the feeling of pleasure in response to a desirable event, and “distress” as the feeling of displeasure in response to an undesirable event. Similarly, “hope” is the feeling of pleasure in anticipation of a desirable event or response to an uncertain desirable event while “fear” is the feeling of displeasure in anticipation of an undesirable event or response to an uncertain undesirable event. They also define “anger” as the feeling of displeasure in response to an undesirable event caused by someone else's action and “disgust” as the feeling of disliking an unappealing object or situation. In contrast, “shame and guilt” are defined as feelings of displeasure in response to one's own blameworthy actions or undesirable events caused by oneself.

Although Ortony et al.'s [8] theory explains some basic emotions, there is still no agreement among different theories regarding the number of basic emotions. For instance, Ortony et al. [8] reject “surprise” as an emotion because they believe it is a cognitive state related to unexpectedness rather than a valenced reaction. Other theorists, including Ekman and Plutchik, take a combinatorial approach, suggesting that primary emotions can combine to produce other emotions. Ekman [10] proposed six basic emotions with distinct facial expressions that are universally recognized: joy, sadness, anger, fear, disgust, and surprise. Plutchik [11] added “trust” and “anticipation” to Ekman's set based on their relationship to adaptive biological processes. This

formulation of human emotions is presented in the famous “Wheels of Emotion” [11] as depicted in the Figure 1. In addition to the eight basic emotions (2nd layer from the center in Figure 1), this diagram shows various other compound emotions and relationships between them. For instance, each of these basic emotions has its polar opposite, located on the opposite end of the diagram (e.g., joy vs. sadness). The emotions in the outer boundary without any background color indicate a combination of two primary emotions. For instance, “anticipation” and “joy” in conjunction form “optimism”.

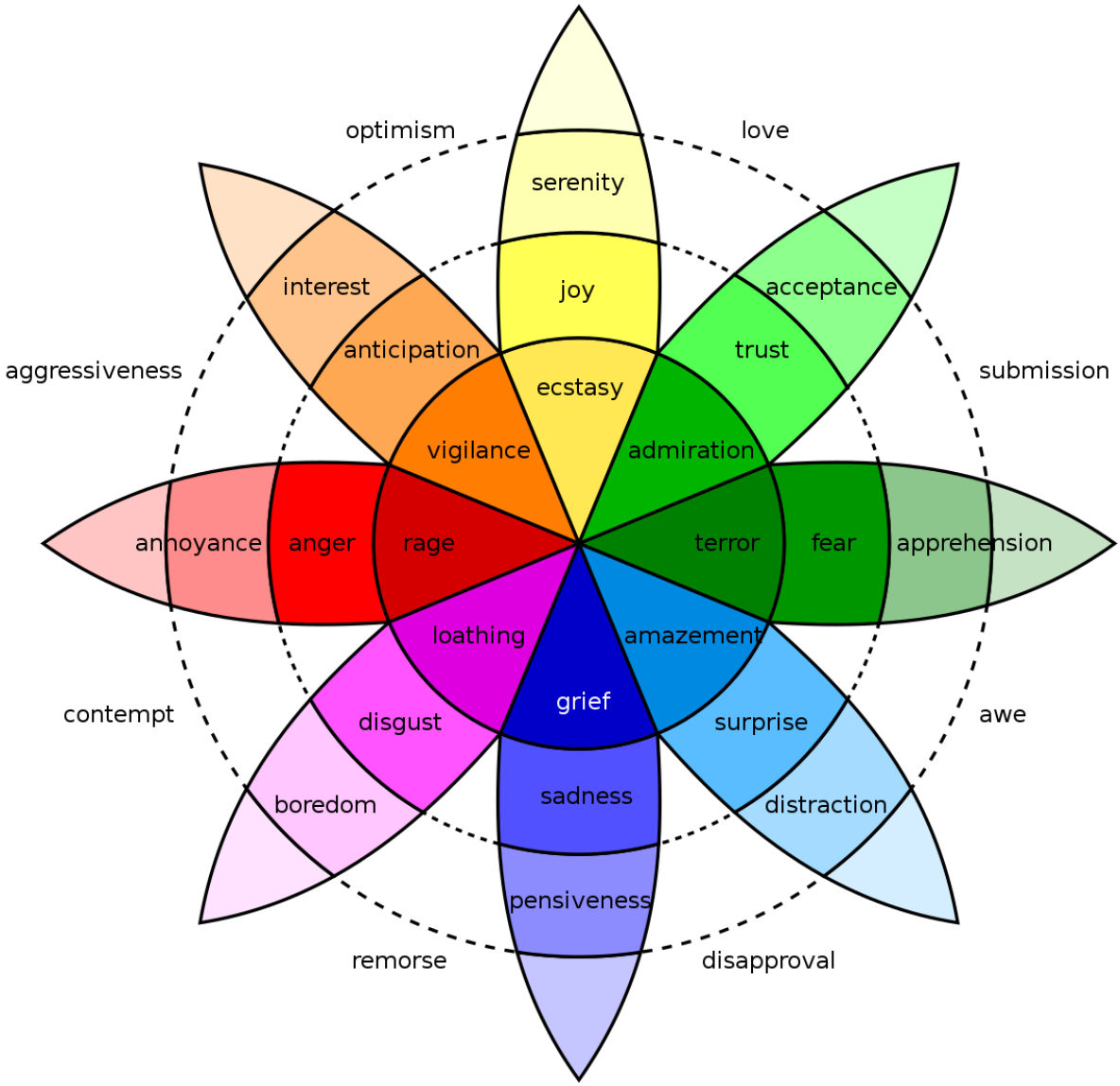

Figure 1: Robert Plutchik's “Wheel of emotion” [11]

Although emotion and sentiment are related ideas and there are meaningful overlaps between them, they are not completely identical or interchangeable as they are often depicted in day-to-day human conversations. Like emotion, sentiment has also received several different definitions in the literature. For instance, Pang et al. [12] suggest that sentiment refers to the attitudes, emotions, and opinions expressed in written or spoken language. In this definition, we observe that sentiment is one of the ways emotions are manifested. It is as if sentiment is a response generated by one or

more emotions. Gross et al. [13] define sentiment as a type of affective state that involves the appraisal of an object, event, or situation in terms of its positive or negative valence. This definition in essence is closer to emotion as defined by Ortony et al. [8] where they propose that emotion is valenced (positive-negative) reactions to events, agents, or objects. Osgood et al.'s [14] definition of sentiment also closely resembles this where they describe sentiment as affective reactions people have towards objects, people, events, and experiences. Some researchers also use the term sentiment and emotion interchangeably though with the acknowledgment that they are not identical [15]. Sentiment usually connects the experienced emotion with an action [7]. Emotions are typically intuitive whereas a person's sentiment towards an entity is more likely to be organized and targeted. On this front, the word "sentiment" is closer to the word "opinion", which means "*a thought or belief about something or someone*" [16]. As opposed to emotion, which can have multiple dimensions (e.g., eight distinct basic emotions), sentiment is typically expressed as binary: negative or positive, and the polar continuum that exists between them. If we express negative sentiment as "-1" and positive as "1", the whole range of real numbers between them represents different sentiment intensity in either direction. At the center, when the value is "0", the sentiment is neither *positive* nor *negative*, or in other word: *neutral*.

In summary, emotion is a complex psychological state that encompasses subjective feelings, physiological changes, and expressive behaviors. It can be triggered by internal or external stimuli and can vary in intensity and duration. In contrast, sentiment refers to a positive or negative attitude, opinion, or feeling directed toward a specific entity, such as a person, product, or event. It is often depicted as a manifestation of emotion. As discussed above, there are overlaps between emotion and sentiment as they both involve affective states and human feelings, however, they differ in their scope and specificity.

2.1.1 Sentiment Analysis

Sentiment Analysis, also commonly referred to as "Sentiment Classification", is a special type of NLP task (Figure 2). Sentiment Classification (or text classification in general) often involves understanding the meaning of the sentence rather than its syntactical structure. This is the reason why it fits well under the "semantic" sub-category within NLP's NLU branch.

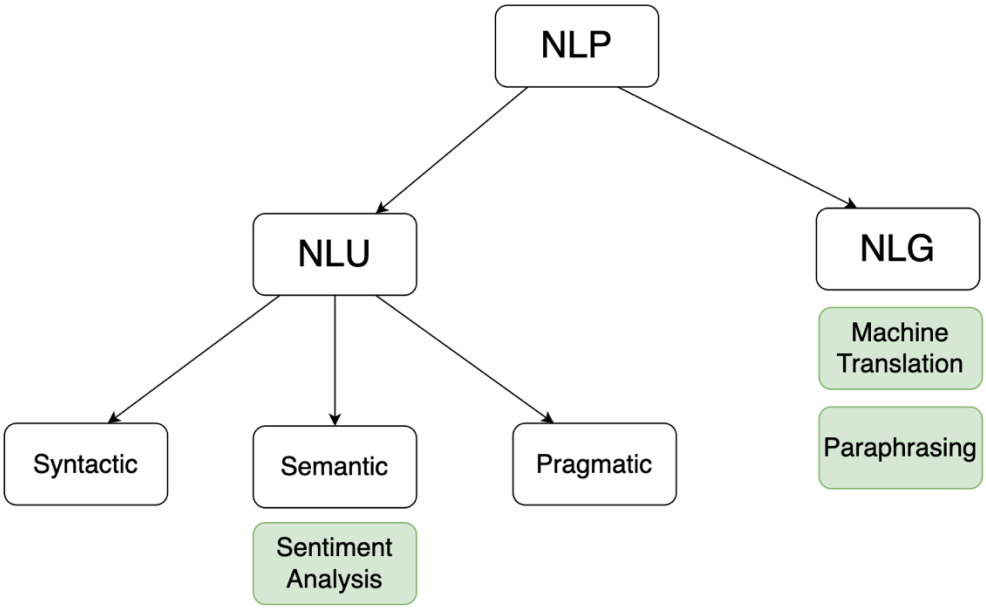

Figure 2: Sentiment Analysis within the NLP taxonomy. Adapted from [1]

Although NLP has a long history of research, little research has been done on Sentiment Analysis before the year 2000. The term "Sentiment Analysis" first appeared in 2003 [17] though the research on the analysis of sentiment appeared earlier in 2000 [18]. Since then, it has been an active and growing research area. The Sentiment Analysis process identifies the sentiment expressed in a given text [19], and then classifies it often in terms of "positive" or negative" sentiment. It is a type of text classification task, where the given text inputs are classified into different sentiment polarity. Dang et al. [20] defined Sentiment Analysis as "*a process of extracting information about an entity and automatically identifying any of the subjectivities of that entity"*. According to Dang et al. [20], the objective of Sentiment Analysis is to determine whether text generated by users conveys their positive, negative, or neutral opinions. Pang et al. [21] define Sentiment Analysis as the "*computational study of opinions, feelings, and subjectivity in text*". According to Alsaeedi et al. [22], "*Sentiment Analysis is a means of assessing written or spoken languages to decide whether articulation is positive, negative or neutral and to what degree*". They elaborate by stating that it alludes "*to the utilization of natural language processing, text mining, computational linguistics, and bio measurements to methodically recognize, extricate, evaluate, and examine emotional states and subjective information*". Sentiments can be denoted as positive, negative, or neutral or

an arithmetical score conveying the strength of the sentiment, often termed as "sentiment polarity score" [23].

Sentiment Analysis can be performed at many different levels, which researchers tend to categorize into three levels:

- **Document level**, where the overall sentiment of an entire document (e.g., a news article) is taken into consideration.
- **Sentence level**, which concerns about the sentiment expressed in a single sentence (e.g., typical textual content from social media). In this thesis, we primarily focus on this level of Sentiment Analysis.
- **Entity and aspect level**, which, in addition to analyzing the sentiment expressed in a given text (typically a sentence), also attempts to discover what entity the sentiment is associated with [15].

Sentiment Analysis also has various subtasks: depending on the number of sentiment classes it can be binary or multi-class; it can also be a single or multi-label classification task depending on the number of sentiments expressed in any given input. There are also further variations of Sentiment Analysis where the aspect of sentiment is also identified (i.e., entity and aspect level Sentiment Analysis). For example, in a product review, aspect-oriented Sentiment Analysis [24] can determine the sentiment as well as the "aspect" of the product that the sentiment is related to, based on the context provided in the given review.

In summary, Sentiment Analysis is a process that often leverages computational power and techniques in order to determine the sentiment expressed in a given piece of text.

2.1.2 Overview of Sentiment Analysis Literature

A typical Sentiment Analysis process starts with data preprocessing. There are various preprocessing techniques that can be applied to the input data before the actual classification task. For example, stop-words removal is a common preprocessing technique. Stop-words, such as, "is", "are", "the", "of", are those that are typically present in abundance in any given text but carry little weight in terms of the overall meaning or sentiment of the sentence. These words are considered noise and are typically removed before performing a Sentiment Classification task. Another common preprocessing technique is the tokenization of sentences into words.

Figure 3 demonstrates an example of the tokenization process. Each word in a given sentence is separated, which are called tokens. These individual tokens are then further processed in different ways (e.g., lemmatization) based on the Sentiment Classification technique in use.

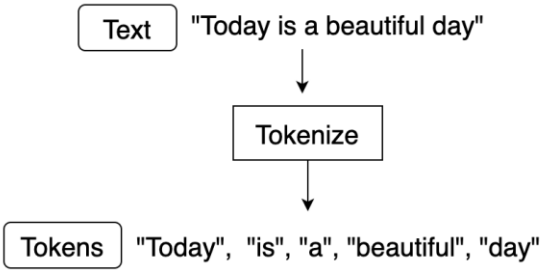

Figure 3: Tokenization of a sentence

After the data preprocessing is done, the actual Sentiment Classification process begins. There are two primary ways to perform the Sentiment Classification task - (i) lexicon-based and (ii) machine learning-based approaches [25]. Machine learning-based approaches can be further grouped into two categories: traditional machine learning and deep learning. Figure 4 represents an overview of the Sentiment Analysis process encompassing various approaches.

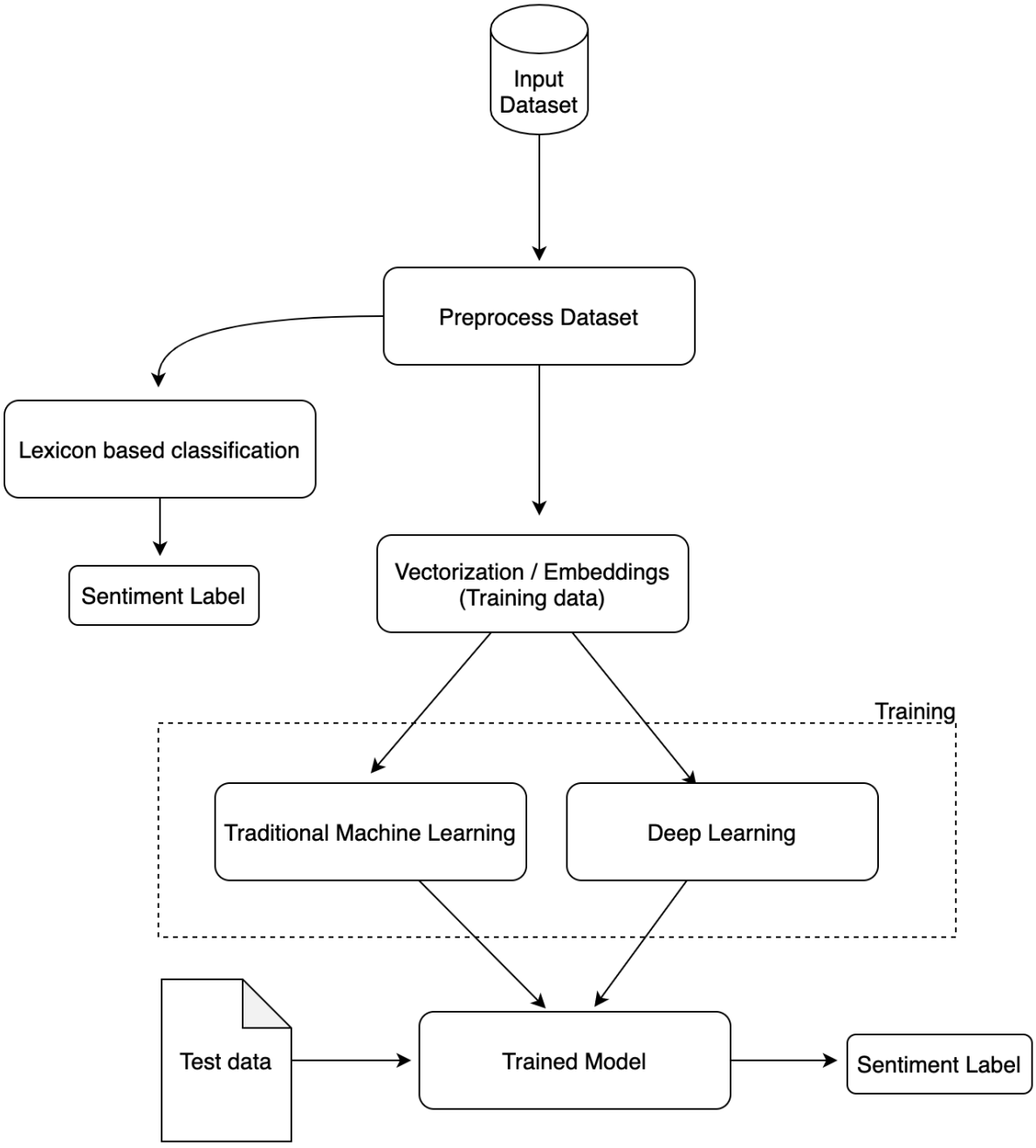

Figure 4: Sentiment Analysis Process Overview

2.1.2.1 Lexicon-based Approach

In the lexicon-based approach, the sentiment of a given text is calculated from the polarity of the words or phrases in that text [26]. A lexicon (i.e., a dictionary) of words with sentiment polarity

assigned to them is used [27]–[29]. Typically, the overall sentiment score of the text is the average of the computed sum of the polarities for each word found in the text. Figure 5 represents a basic example of a lexicon-based Sentiment Analysis process. There is a total of nine words. Only two words among them contribute to the sentiment of the overall sentence: “exciting” and “expensive” as they are ones present in the lexicon being used. Figure 5 shows a part of the lexicon where we have these “sentiment words” along with their sentiment polarity score. By adding up the polarity score of each sentiment word for a given sentence, we can find the overall sentiment of the sentence. In this case, for example, the sum is: +5 + (-3) = +2. This means the overall sentiment of this sentence is positive.

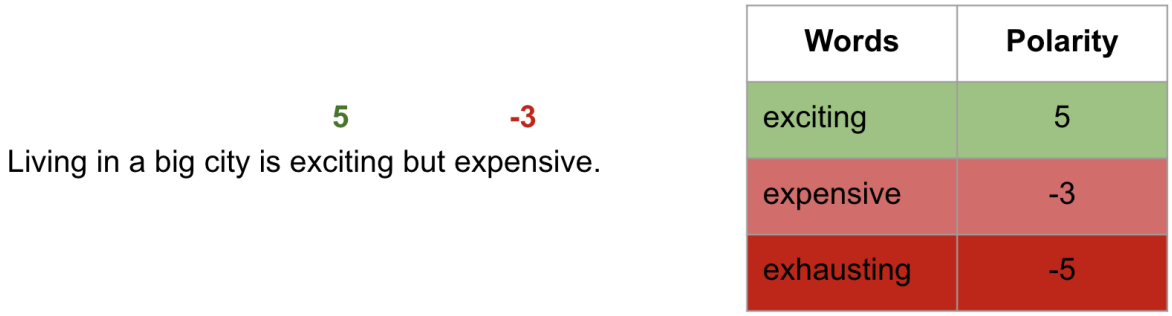

Figure 5: Sentiment Analysis using a lexicon

Lexicon-based methods use lists of words (i.e., dictionary) marked by polarity score to determine the overall opinion score of a given text input. Their main advantage compared to machine learning methods is that they do not require training data. There are two common ways to build a sentiment lexicon: existing lexicon-based construction method that artificially creates a lexicon, and automatic or semiautomatic construction method [25]. General Inquirer (GI) is considered the earliest emotional thesaurus and affective analysis program. It derives emotional words from Harvard IV-4 Dictionary [30] and Lasswell’s Dictionary [31]. GI labels each word with polarity, strength, and parts of speech. These labels make it useful for Sentiment Analysis tasks. Typically, GI takes a pre-set list of positive and negative words and analyzes the polarity of documents based on the frequency of words from each category. This type of lexicon-based method is time-consuming and contains less vocabulary. Automatic or semiautomatic construction of sentiment lexicon later became more dominant. For instance, SentiWordNet [28] is a lexical resource for

Sentiment Analysis, which assigns each entry of WordNet [32] three sentiment scores: positivity, negativity, and objectivity. SentiStrength [33], a lexicon-based algorithm subdivides human emotion into more dimensions. It estimates the strength of positive and negative sentiment in short texts, even for informal language. This makes it well-suited for social media text. SentiStrength reports two sentiment strengths: −1 (not negative) to −5 (extremely negative), 1 (not positive) to 5 (extremely positive).

Ortega et al. [34] proposed a three-step technique for Sentiment Analysis using Twitter data. The first step was pre-processing and the second and third steps were polarity detection and rule-based classification consecutively. The latter two steps leveraged WordNet and SentiWordNet. Their approach demonstrated promising results when evaluated on the SemEval-2013 Task-2 dataset [35]. Saif et al. [36] developed another lexicon-based approach for social media Sentiment Analysis, particularly Twitter, named SentiCircles. By considering the patterns of words that co-occur in different contexts, it updated the pre-assigned scores and polarity of words in sentiment lexicons. Their evaluation demonstrated that the methods based on SenitCircles outperformed methods based on SentiWordNet.

A technique to expand the list of sentiment words in a lexicon is to leverage semantic relationships such as synonyms and antonyms. A typical scenario is to define a small number of sentiment words, often annotated manually, and then to enlarge this initial list by inserting words with similar semantics [37]. The limitation of this approach is that the extension of the opinion information is restricted and dependent on the initial list of seed words. To address this, Feng et al. [38] suggested connotation lexicons enclose subtle dimensions of a word's sentiment. After defining a list of seed words, they used graph-based algorithms (e.g., PageRank [39]) to learn the connotation lexicon together with the connotative predicates. They evaluated their approach in SemEval-2007 Task-14 [40] demonstrating promising results.

In an aspect-based Sentiment Analysis study, Salas-Zárate et al. [41] used semantic annotation to identify aspects concerning diabetes in tweets. Subsequently, they used these identified aspects to perform aspect-based Sentiment Analysis using SentiWordNet. Their results revealed that for this aspect-based Sentiment Analysis task, the "N-gram around" method (i.e., N-gram [42] words before the aspect and after the aspect) obtained promising results. The author noted that their results could be improved by using domain-specific sentiment lexicons and by using an automatic

or semi-automatic approach to create ontologies in order to obtain more domain-related knowledge in the ontology.

Some researchers attempted to generate dynamic lexicons for a particular domain based on the domain data and existing static lexicons. For instance, Mowlaei et al. [24] developed two dynamic sentiment lexicon generation methods: one is based on the frequency and some data pre-processing techniques such as negation, and the other one is based on a genetic algorithm. Their evaluation suggests that these dynamically generated lexicons, which are essentially a product of the fusion between existing static lexicons and domain-specific information from the given dataset, performed significantly better than static lexicons.

Although recent lexicon-based approaches have seen improved efficiency in Sentiment Classification, it still falls short in several areas. For instance, it cannot detect linguistic nuances such as irony or sarcasm. It typically fails or underperforms in capturing the context of the language and it is significantly outperformed by machine learning approaches [25].

2.1.2.2 Machine Learning-based Approach

Machine learning algorithms are used for many NLP tasks including Sentiment Analysis. The most common machine learning approach used for Sentiment Analysis on textual data is supervised learning. By using a large number of labeled input data for training, supervised learning-based models can classify the sentiment of a given text. Go et al. [43] conducted one of the first studies to utilize a supervised machine learning-based approach to analyze sentiment in social media texts. They classified the tweets as either positive or negative. To avoid manual tagging of sentiment, they used distant supervision (i.e., using existing labeled data or knowledge to automatically label large amounts of unlabeled data for a given task) to build a machine-learning classifier as demonstrated in [44]. They differentiated negative and positive tweets with emoticons within the Tweets. They developed a large training data set consisting of 1,600,000 tweets. They inspected NB, MaxEnt, and SVM classifiers on this training data. Their choice of classifier algorithms was inspired by Pang et al.'s [45] work who performed Sentiment Analysis of movie reviews. Bigrams [42], unigrams [42], and POS tags were used as features. Their evaluations suggest that NB with bigrams as features outperformed other classifiers and achieved an accuracy of 82.7%. They also concluded that adding negation as an explicit feature with unigrams and using POS tags are not useful for polarity classification.

Pak and Paroubek [46] also used emoticons as labels to annotate about 300,000 tweets but as a non-binary multiclass classification task by classifying the tweets as positive, negative, or neutral. They evaluated the performance of SVM, MNB (i.e., Multinominal NB), and Conditional Random Field (CRF) [47] using various features including unigrams, bigrams, n-grams, and the position of n-grams. MNB with n-grams and POS tags demonstrated the best result, with an observed increase in performance with more training data. Another contribution of the work included the identification of the co-occurrence of the pronoun for the first person and the adjectives in the opinionated messages. Though they included a neutral category, which is an improvement over Go et al's [43] work, labeling their training dataset followed a similar procedure as Go et al. which heavily relied on emoticons. Since emoticons may not always represent the actual sentiment expressed in a tweet, it is likely to generate a considerable amount of inappropriately labeled data.

Barbosa and Feng [48] presented a two-step classifier. The first step determined whether the message was opinionated or not. The second step aimed to further classify the sentiment of the tweet as positive or negative. They created a training dataset with 200,000 tweets annotated automatically with multiple sentiment detection tools. In their experiment, the SVM classifier demonstrated the best result with an accuracy of 81.9% for subjectivity detection and 81.3% for sentiment polarity detection. One of the limitations of this study, as acknowledged by the authors, was that the classifier was not robust when analyzing sentences with antagonistic sentiments.

Jiang et al. [49] experimented with including target-dependent features along with usual target-independent features in their Sentiment Analysis work. The author provided an example sentence to explain the importance of target dependency in the following tweet – "Windows 7 is much better than Vista!". In this tweet, though the sentiment is positive towards Windows 7, it is negative towards Vista; and a typical target-independent classifier ignores this fact and is likely to classify the tweet as positive. The authors manually defined rules to detect the syntactic patterns that determined a term's relation to a specific object. They used many features from Twitter such as retweets, replies, and mentions to create a graph that reflects the similarities of tweets. They used SVM for subjectivity and polarity classification. Their experiment concluded that utilizing target identification improved the accuracy of Sentiment Classification, achieving an accuracy of 85%.

More recently, a group of researchers proposed a method to improve SVM classification accuracy by sub-setting training data using clustering [50]. The clustering-based approach involved an

instance selection method using data points with maximum, minimum, and average distances to each cluster center. Their results showed the higher accuracy of the applied method compared to earlier work [51] by the same authors which did not utilize this clustering method to subset training data.

Bania et al. [52] evaluated the performance of four traditional non-parametric machine learning models [53] while analyzing public sentiment expressed in tweets regarding the Covid-19 pandemic [54]. The learning algorithms included Naïve Bayes (NB), Random Forest (RF), and Support Vector Machine (SVM). Two variations of the NB: Gaussian-NB (G-NB), and Bernoulli's-NB (B-NB). They leveraged the TF-IDF feature extraction scheme with unigram, bi-gram, and tri-gram techniques. Their result suggests that RF and B-NB models outperform the other two models. They also revealed that linear-SVM based models had higher computation costs. Though this study did not propose a new technique, it indicates the generalizability of Sentiment Analysis techniques across domains, especially when dealing with similar data sources. This generalizability, while desirable, may not always contribute to the expected accuracy. The authors also considered the fact that many of the tweets used in the experiment might be coming from bots or fake accounts, especially since the study analyzed tweets regarding a globally trending topic on Twitter.

Some researchers attempted to apply unsupervised learning algorithms in classifying sentiment. These algorithms are used to analyze and cluster unlabeled datasets, which discover hidden patterns or data groupings without the need for human intervention or data labels. Some researchers have attempted to apply it to Sentiment Analysis. For instance, Riaz et al. [55] performed Sentiment Analysis on the customer review data using unsupervised learning. Their analysis was at the phrase level which aimed to understand customer preference by analyzing subjective expressions. The researchers calculated the strength of sentiment words to determine the intensity of each expression. They applied k-means clustering for categorizing the words in various clusters based on their intensity. Fully unsupervised methods like this typically suffer from low accuracy in classification tasks similar to Sentiment Analysis. However, using only a few labeled data, the semi-supervised learning method, which is a sub-class of supervised learning can achieve reasonably better accuracy in Sentiment Classification. This often includes generating "pseudo labels" for the unlabeled training data in iterations, primarily based on the few initial labeled data. For instance, Iosifidis et al. [56] leveraged a semi-supervised approach to annotate

large datasets with sentiment labels using techniques such as self-learning and co-training. They observed that, unlike co-training, self-learning propagates the original class imbalance to successive iterations. They noticed that in initial iterations with a very low number of labels, co-training performed better than self-learning; however, in later iterations, self-learning improves faster than the co-training technique.

One study demonstrated how lexicon and semi-supervised methods can be combined to offer better Sentiment Classification [57]. They integrated several sentiment lexicons to generate a unified solution that provides a high-quality lexicon. For semi-supervised classification, they generated a high-quality data set that is smaller than the one used in traditional supervised learning. They leveraged confidence levels to fine-tune their classifier. Eventually, they developed an architecture that combines their high-quality lexicons and high-quality training dataset in semi-supervised algorithms, which demonstrated comparable performance with fully supervised methods.

An issue with supervised machine learning techniques to perform Sentiment Analysis is that they require a large amount of training data. We observe in the literature that some researchers have attempted to mitigate this issue by incorporating semi-supervised methods. We also notice that there is a growing number of studies that combine various algorithms to form one solution, as well as solutions that combine machine learning-based approaches with lexicon-based ones. More recently, deep learning-based methods to perform Sentiment Analysis have garnered popularity among researchers.

Deep learning is one of the fastest-growing sub-domains of machine learning. It has been seeing extensive usage in solving perceptual problems such as image recognition and understanding natural languages. Deep learning uses neural networks to learn many levels of abstraction. In text-related tasks, deep-learning approaches typically include two steps. First, they learn word embeddings from the text collection, and these are then applied to produce the representations of the documents. Tang et al. [58] and Zhang and Zheng [59] are among the first research groups to apply deep learning approaches for Sentiment Analysis. Both groups experimented with part of speech (POS) as a text feature and TF-IDF to calculate the weight of words for the analysis. Zhang et al. also compared Extreme Learning Machine (ELM) (with kernels) [60] with SVM and found that ELM significantly outperformed SVM in terms of classification accuracy. Since then, several studies applied deep-learning-based Sentiment Analysis in different domains, including finance

[61], weather-related tweets [62], trip advisors [63], and movie reviews [64], [65]. In many of these studies [61], [62], input data was transformed into word embedding using tools such as Word2Vec [66] in order to classify the sentiment of text using deep learning algorithms.

A group of researchers [67] proposed combining sentiment and semantic features in a Long Short-Term Memory (LSTM) model (SS-LSTM) based on emotion detection. They detected emotions such as happy, sad, and angry in conversational pairs on Twitter. They applied semi-automated techniques to gather a dataset containing over 17 million tweets. Their approach involved combining sentiment and semantic features from user utterances using Sentiment Specific Word Embedding (SSWE) [68] and GloVe embeddings [69] respectively without requiring any hand-crafted features. Their results suggest that the proposed SS-LSTM model outperforms traditional machine learning baselines as well as other off-the-shelf deep learning models.

Li et al. [70] studied the impact of data quality on Sentiment Classification performance by considering three criteria: informativeness, readability, and subjectivity. Their dataset consisted of online product reviews, and they applied three deep learning techniques to this data: Simple Recurrent Network (SRN), LSTM, and CNN. Their evaluation suggested that two factors affect the level of accuracy of Sentiment Analysis: readability and length of the reviews. Higher readability and shorter text datasets yielded higher-quality Sentiment Classification.

Dang et al. [20] applied deep learning models (DNN, RNN, CNN) with TF-IDF and word embedding to Twitter datasets and implemented a Sentiment Analysis technique with a special focus on the problem of sentiment polarity analysis. The results suggest that combining deep learning techniques with word embedding than with TF-IDF when performing a Sentiment Analysis can yield better results. Their experiments revealed that CNN outperforms other models considering both accuracy and processing time. Though RNN's reliability is slightly higher than CNN with most datasets, it requires significantly more processing time. This is because RNNs are designed to handle sequential data by processing one element at a time while maintaining a hidden state. This sequential processing can result in higher accuracy for certain text classification tasks, as RNNs are able to capture contextual information and long-term dependencies. However, this sequential processing also makes RNNs more computationally expensive, as each element in the sequence must be processed sequentially. This can result in longer processing times, especially for

longer sequences or larger datasets. On the other hand, CNNs can process multiple elements in parallel, making them faster and more efficient for certain types of text classification tasks.

Jebbara et al. [71] leveraged transfer learning while experimenting with cross-lingual Sentiment Analysis where the focus was on supporting Sentiment Analysis in languages that are under-resourced in terms of training data. For this approach, they leveraged zero-shot transfer learning using pretrained multilingual word embeddings. While their best achieved F1-score remained low (under 0.60), they have demonstrated performance gains over other approaches attempting to address the same problem.

Similar to other applications of machine learning, researchers have experimented with combining machine learning and deep learning approaches within one architecture in order to build a more performant Sentiment Classification method. For instance, Araque et al. [72] created a deep learning-based sentiment classifier using a word-embedding model and a linear machine learning algorithm which outperformed both individual classifiers.

Another study produced a performant deep learning model for Sentiment Analysis which is based on an attention-based bi-directional CNN-RNN deep network that leverages two independent LSTM and (Gated Recurrent Unit) GRU layers [73]. They used GloVe word embedding vectors as the initial weights of the embedding layer. On top of that, the bidirectional LSTM and GRU networks were used to extract both past and future contexts. The final layer included a typical dense fully connected layer with a Sigmoid activation function to transform the vector into sentiment representation in order to perform the binary sentiment polarity classification.

Deep learning-based methods for Sentiment Analysis have seen increasing popularity in the last few years. There is also the trend of combining multiple algorithms and approaches to achieve better classification accuracy, including a combination of traditional machine learning and deep learning-based techniques. More recently, there has been a number of Sentiment Analysis work that leveraged transfer learning, especially in scenarios where the number of labeled training data was limited.

2.2 Emotion Cause Analysis (ECA)

Sentiment Analysis, which we discuss in the previous section, primarily focuses on categorizing a given text based on sentiment polarity. In contrast, ECA tasks focus on the cause behind the

emotion – it concerns with the question of why a person is experiencing a particular emotion in a given context. One of the earliest works in this area refers to the event, person, or state that elicits an emotional response in the experiencer as the "emotion stimulus" [7]. According to this study, an “experiencer” has a particular emotional state which is often described by the stimulus that elicits it. There can also be a circumstance under which the emotional response occurs or a reason why the stimulus elicits the particular emotional response in the experiencer. For example, in the sentence, “Rizvee is very happy to get the latest smartphone as a gift”, "Rizvee" is the experiencer, "happy" is the emotion, and "latest smartphone as a gift" is the emotion stimulus. Ghazi et al. [9] observe that among the elements of emotion [7], the emotion stimulus can be characterized as the “cause” behind the emotion felt by the experiencer. Learning about the “cause” behind certain expressed emotions is of vital value in many real-life scenarios such as getting a deeper understanding of consumer behavior, and the root cause of public opinion on certain matters. Recognition of this inspired many research works that focus on determining the cause of the emotion in various ways, which form a particular domain within NLP, known as Emotion-Cause Analysis (ECA).

There are various subtasks within ECA. The primary one is Emotion cause extraction (ECE), which is the earliest and most popular ECA task investigated by researchers so far.

The ECE task involves finding the potential causes that lead to expressed emotions in text. A simple representation of this task is to formulate it as a binary classification problem at the clause level [74]. The aim is to determine, for each clause in a document, whether it is the cause of the annotated emotion. In the ECE task, the input is a document and the annotated emotion, and the goal is to extract the cause clause that corresponds to the given emotion.

Emotion-Cause Pair Extraction (ECPE) is a newer variation of the ECE task, which focuses on finding all possible pairs of emotions and their corresponding causes within a document [75]. Unlike ECE, the output of the ECPE task is a pair of emotion-cause, without the need for prior annotated emotions. Emotion-Cause Span-Pair extraction and classification (ECSP), is another variation of the ECE task that builds upon ECPE [76]. The goal of this task is to extract the potential span-pair of emotions and their corresponding causes in a document and to classify the emotions for each pair. Thus, Emotion Cause Extraction (ECE) and Emotion-Cause Pair Extraction (ECPE) can be seen as two specific cases of ECSP at the clause level [76].

Figure 6 shows an intuitive example of these ECA tasks. For a given set of clauses contained within one single datapoint, the task of ECE is to identify the clause that contains the emotion-cause. In Figure 6, the ECE task thus aims to identify clause 3 while the emotion "happy" is given. In contrast, the ECPE task identifies the clause that contains the expression of emotion as well as the cause of the expressed emotion, without any given annotated emotion. In this case, it is the identification of the pair (clause 4, clause 3). For the ECSP task, the aim is to identify the emotion expression and emotion cause at a more granular span level instead of the entire clause. As a result, the expected output would be to extract the specific spans within clause 3 and clause 4 (highlighted in bold in Figure 6 for clarity), as well as the classification of the emotion (i.e., "happy" in this example).

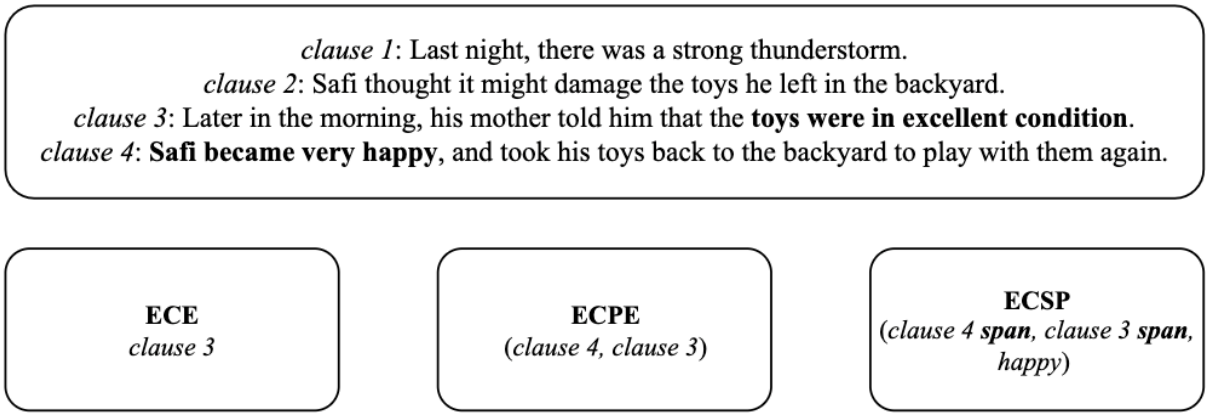

Figure 6: Different ECA Tasks

2.2.1 Overview of Emotion-Cause Analysis Literature

ECA is relatively a newer research area compared to Sentiment Analysis. With a few initial exceptions, most of the ECA-related tasks are performed with machine learning techniques. There are a few rule-based techniques for the ECE task. Apart from these, most other ECA subtasks leverage traditional machine learning technologies, and more recently, deep learning techniques.

2.2.1.1 Rule-based Approach

The first ECA task was proposed by Lee et al. [77] where they manually created a small corpus of emotion causes, using the Academia Sinica Balanced Chinese Corpus. They introduced the emotion cause extraction (ECE) task, which was defined as a word-level sequence labeling

problem, with both emotion expression and emotion cause annotated. Some other early studies also explored different rule-based approaches to extract emotion causes from a given text. For example, Li and Xu [78] created an emotion cause corpus leveraging Chinese microblog posts and proposed a rule-based method to extract emotion causes by incorporating knowledge from other fields such as sociology. Leveraging the same corpus, Gao et al. [79], [80] then designed a set of complex rules considering a cognitive emotion model and emotions categories to extract emotion causes (some other rule-based examples can be found at [81], [82]). In its most basic form, these rule-based methods for the ECE tasks concentrate on detecting explicit emotions expressed through emotion keywords. They annotated each emotion keyword with its corresponding cause. These studies primarily considered that most emotion causes can be found within the same clause as the expression of emotion. Based on this, they conclude that a clause may be the most suitable unit for detecting a cause. The clauses were identified through punctuation such as commas, periods, question marks, and exclamation marks. The authors utilized linguistic cues such as causative verbs and perception verbs to create patterns for extracting general cause expressions or specific constructions for emotion causes. They formalized the emotion cause detection as a multi-label classification problem, where each instance could have more than one label, such as "left-1, left-0", to indicate the location of the clauses that are part of the cause. The method also involved keyword spotting to identify the emotion word in the sentence and find its cause.

In later years, more advanced ECA systems have been developed, employing Conditional Random Fields (CRFs) as a probabilistic framework for identifying the cause of emotions. These systems have been trained on annotated datasets containing cause-spans, resulting in accurate identification of emotional causes [9]. Recent research studies in the ECA domain leverage machine learning techniques, primarily using supervised learning. The outcome of the task can vary based on the type of ECA subtasks. This difference in outcome is also reflected in the type of machine learning techniques and structure of the training dataset used in the study. We discuss them below.

2.2.1.2 Machine-learning based Approach

Gui et al. [9], [83] built a microblog emotion cause corpus based on the NLPCC 2013 emotion analysis task and proposed a machine learning method using SVMs and CRFs to extract emotion causes. These studies regarded the ECE task as a sequence labeling problem. Gui et al. later released a Chinese emotion cause corpus from a public SINA city news and proposed a multi-

kernel-based method for emotion cause extraction [84]. Unlike previous corpora, the ECE task in this corpus was defined as a clause classification problem, where the goal is to predict if a clause in a document is an emotion cause. It was evaluated using clause-level precision, recall, and F1 score metrics, and has since become a benchmark dataset for ECE research. Several other techniques for ECE were approached based on the traditional machine learning approach (e.g. [85], [74]).

In recent years, deep learning techniques have also been applied to perform the ECE task, including LSTM [86], deep memory network [87] etc. Most of these studies treated the ECE task as a set of independent clause classification tasks, ignoring the relationships between clauses in a document. To address this, Ding et al. [88] converted the task to a reordered clause classification problem, where predictions of previous clauses were used as features for predicting subsequent clauses. In contrast, another study [89] proposed a joint emotion cause extraction framework that models and classifies multiple clauses in a document simultaneously leveraging Transformer as the clause-level encoder to model the relationships between multiple clauses.

Recently, researchers suggested that the ECE task has two limitations: (1) it needs the emotion annotation, and (2) it does not take into account the mutual connection between the emotion clause and cause clause [90]. In order to address these limitations, emotion–cause pair extraction (ECPE) task was proposed, which extracts both the emotion clause and cause clause and devises an emotion–cause pair [75]. This, however, conducts the entire process in a two-step mechanism. First, it extracts all clauses – potential clauses containing the emotion expression and clauses containing the cause of the expressed emotion. Second, it matches these clauses to form the emotion–cause pair. This type of two-step method generally causes error propagation from the first step to the second. To resolve this, a group of researchers proposed a unified architecture that extracts the emotion-cause clause pair using a ranking system [91]. Some subsequent work treats the ECPE task as a sequence labeling problem. For instance, one study treated the relative distance between the emotion clause and cause clause as a part of the label [92]. Another study reformulated the ECPE task as a unified sequence labeling task which extracts all possible pairs of emotion-cause [93]. An interesting recent approach tackled the ECPE task as a dual question-answer task [94], leveraging the attention network. It considers the emotions and causes as two separate questions and extracts answers from the given context separately leveraging the attention network [95]. Another study proposed a novel technique by incorporating the distances between the

emotion and cause clause into a novel tagging scheme [96]. They perform the ECPE task by adjusting the paired tagging distribution based on the estimated distribution of the auxiliary tasks.

Inspired by recent success in the span-based models for co-reference resolution and syntactic parsing [97], [98], Bi et al. [76] proposed a new task within ECA, named Emotion-Cause Span-Pair extraction and classification (ECSP). This is also a variation of the original ECE task with more advanced requirements. The objective of this task is to extract the potential span-pair of emotion and related causes in a given document and classify emotion for each pair. This work focuses on finer granularity in terms of detecting the actual cause of the emotion, compared to the previous works where the emotion clauses were detected, and these clauses often contained unrelated information. The main idea is to annotate each emotion and cause with its span boundary and emotion categorization. This annotation leads to the creation of a span-based extract-then-classify model, which directly pairs emotions and causes from the document while following the target span boundaries. The categorization is then determined through the use of pair representations and localized context. This method is advantageous as it allows for a consistent interpretation of clause-based and span-based tasks. Furthermore, the polarity is determined based on the targeted span representation, ensuring that all target words are taken into account before making predictions and avoiding sentiment inconsistencies.

Compared to Sentiment Analysis, ECA is a relatively new research area that initially relied on rule-based techniques and slowly evolved into an NLP domain that primarily leverages supervised machine learning techniques. A variety of subtasks have been included in this domain over the years, which all primarily focus on emotion classification and extraction of the emotion-cause in a given text.

2.3 An Overview of Related Machine Learning Technologies

In this section, we offer a brief overview of machine learning techniques and concepts that are relevant to this thesis.

2.3.1 Representation of Text in Machine Learning

Machine learning algorithms typically process data in the form of vectors. Translating textual data to vectors may not be always straightforward and how it is done impacts the overall outcome of

the machine learning algorithm. We discuss some of the common techniques for representing textual data in machine learning algorithms below:

One-hot vector: A common technique to represent text is using a sequence of discrete tokens where each of these tokens is denoted by a one-hot vector. One-hot vector is a large sparse vector with a single dimension for each word in the overall data. This representation method is popular for categorical data such as the ones used for Sentiment Analysis (i.e., different sentiment polarity representing each category).

Bag-of-Words: Bag-of-Words (BoW) is a similar concept to the one-hot vector representation of texts but with the addition of frequency (of words) in the representation whereas one-hot vector only contains information about the presence (1) and absence (0) of words. Let us consider the following three texts as an example:

- Text 1: The weather is bad.
- Text 2: The weather is good.
- Text 3: The weather is unpredictable today, but the afternoon is sunny.

We first develop a vocabulary consisting of all the unique words in the above texts: 'The', 'weather', 'is', 'bad', 'good', 'unpredictable', 'today', 'but', 'afternoon', 'sunny'. This is our BoW. Now we can represent our three texts using BoW in the following manner:

	The	**weather**	**is**	**bad**	**good**	**unpredictable**	**today**	**but**	**afternoon**	**sunny**	**Word count**
Text 1	1	1	1	1	0	0	0	0	0	0	**4**
Text 2	1	1	1	0	1	0	0	0	0	0	**4**
Text 3	2	1	2	0	0	1	1	1	1	1	**10**

Table 1: Vectorization Example

Based on the above table, the following are the vector representations for our texts using the BoW approach:

- Text 1: [1, 1, 1, 1, 0, 0, 0, 0, 0, 0]
- Text 2: [1, 1, 1, 0, 1, 0, 0, 0, 0, 0]
- Text 3: [2, 1, 2, 0, 0, 1, 1, 1, 1, 1]

These vector representations are now ready to be used by machine learning algorithms. However, since this method requires having a separate dimension for every single word in BoW, the one-hot

vector representation can become extremely large. This may lead to a computationally unplausible situation when the text in question contains hundreds or thousands of words. Some researchers try to put a "fixed" size to the BoW in order to reduce the vector size for each text. However, this creates the risk of losing important information that may be of significance for Sentiment Analysis. Another issue with one-hot vectors, in general, is that any two words are orthogonal regardless of the similarity of words, lacking important information regarding the meaning of associated words.

TF-IDF: Some of the limitations of BoW can be avoided by using Term Frequency-Inverse Document Frequency (TF-IDF). It is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus [99]. Term Frequency (TF) measures how frequently a term occurs in a document. TF is often divided by the document length (i.e., the total number of terms in the document) in order to perform normalization to account for the varying document sizes. On the other hand, Inverse Document Frequency (IDF) measures how important a term is. While computing TF, all terms are considered equally important. However, many words such as "is", "of", "this" contribute little weight to the meaning while appearing frequently in any given text. IDF measure weighs down the frequent terms and scales up the rare ones.

Consider Text 3 in our above example which contains 10 words. The term "is" appears 2 times there. So, the TF measure for the term "is" is: 2/10 = 0.2. We have a total of 3 texts in the above example (Text 1, Text 2, and Text 3), and "is" appears in all of them. This means our IDF count would be: 3 / 3 = 1. Consequently, the overall TF-IDF measure if the product of TF and IDF: 0.2 * 1 = 0.2

Word Embeddings: Word embeddings (also known as "word vectors") is another technique for textual representations, typically leveraged by deep learning approaches. Words in word embeddings are represented with dense vectors in a latent space that better captures the meaning of words [71]. In word embeddings, similar words have a similar encoding. Importantly, we do not need to specify this encoding by hand. It is typically a dense vector of floating-point values where the length of the vector is a parameter that can be specified [100]. The values for embeddings are trainable parameters, meaning they are learned by the model during training. A higher dimension word embedding can contain a fine-grained relationship between words. There are various techniques to generate word embeddings. One example is Continuous Bag of Word (CBOW). In CBOW, the context of a word is represented by multiple words for a given target

word. For example, let us consider text 3 again: “Text 3: The weather is unpredictable today, but the afternoon is sunny”. Here, we can use “unpredictable” and “sunny” as context words for “weather” as the target word. These learned relationships of target words and context words are used by CBOW [101] to create word embeddings that are rich in the syntactical meaning of words in a given context. Another technique is Skip-gram [101], which is the opposite of CBOW. Skip-gram technique attempts to predict the “context words” (*w(t-2)…..w(t+2)* in Figure 7) for a given “word” (depicted as *w(t)* in Figure 7). For this algorithm, the input is one word, and the outputs are the associated context words of that one word. For our Text 3 example above, Skip-gram will output “unpredictable”, and “sunny” as context words for the term “weather” as input. Google Word2vec [66] algorithm uses either CBOW or Skip-gram in order to produce word embeddings.

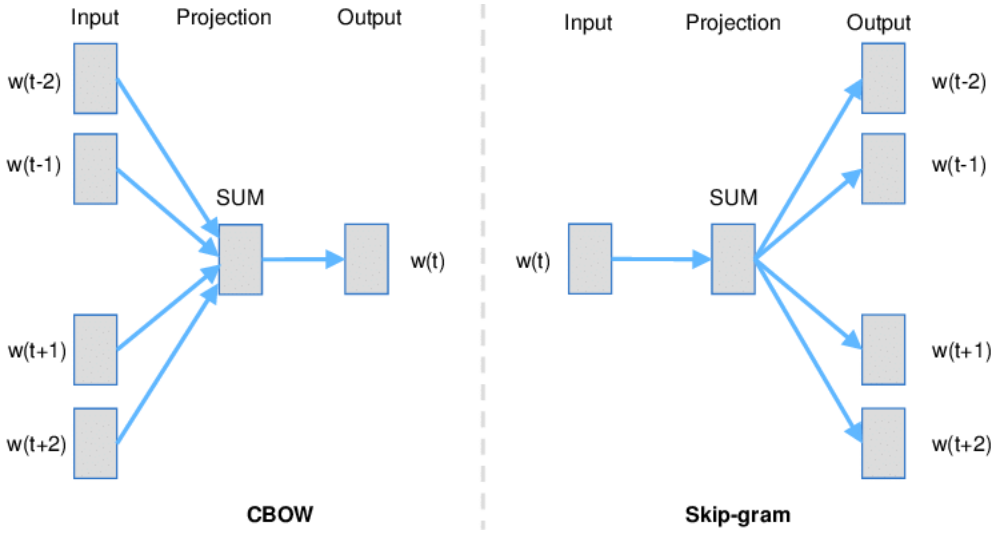

Figure 7: CBOW and Skip-gram [101]

Common word embeddings such as Word2vec are created with shallow neural networks (i.e., neural networks containing only 1 or 2 hidden layers. See [102] for more details.). Typically, this neural network will take one-hot vector representations of words (i.e., sparse matrix) and then use it for unsupervised training of a neural network. Instead of using the output of the neural network, the weight matrix of the fully connected dense hidden layer is stripped out of the model. This layer represents the one-hot vector encoding of words as a dense vector of floating points, which is what is used as word embeddings for the word that one-hot vector input represents.

Word embeddings overcome many common problems of one-hot vector and similar frequency-based vector encodings such as high computational cost and lack of semantic relationship

information of words within a context. By preserving rich contextual information, word embeddings can boost the performance and accuracy of many NLP tasks such as Sentiment Analysis, especially for deep neural network algorithms-based solutions.

2.3.2 Machine Learning Techniques

Artificial Intelligence (AI) enables computers to mimic human intelligence. Machine learning is a subset of AI that enables machines to use the experience to improve at specific tasks [103]. The learning process generally includes feeding data into an algorithm, using the data to train a model, evaluating and iteratively improving the model, and then finally using the model for the desired task. Data used for machine learning is typically divided into training and test dataset. A common training and test data ratio is 80:20. Specific portion of training datasets sometimes includes a third set, called a validation set, which is typically used for cross-validating the machine learning models and adjusting model parameters to find their best values. This step is also known as hyperparameter tuning [104].

Machine learning approaches are commonly divided into two groups: "supervised" and "unsupervised". In a **supervised learning** model, the algorithm learns on a large amount of labeled training datasets. On the other hand, in an **unsupervised** model, unlabeled data is provided that the algorithm attempts to understand by extracting features and patterns on its own [105]. For classification tasks like Sentiment Analysis and Emotion-Cause Analysis, supervised learning methods are more commonly used. Traditional supervised machine learning approach typically involves analyzing previously labeled data, extracting features that model the differences between different classes, and inferring a function that can be used for classifying unlabeled data. Supervised learning also has a subset called "**semi-supervised learning**". It combines a small amount of labeled data with a large amount of unlabeled data during the training process. Semi-supervised learning falls between unsupervised learning (with no labeled training data) and supervised learning (with only labeled training data) [106]. Unlabeled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement in learning accuracy. Labeled training data scarcity is a common issue for machine learning projects as labeling a large amount of data can be expensive and tedious. This is especially true for domain-specific tasks such as Sentiment Analysis. Semi-supervised learning algorithms have been historically proven useful in such scenarios (e.g., [107]).

Recently, **deep learning**, which is a subset of machine learning, is gaining rapid adaption for many NLP tasks. There are several fundamental differences in how a typical machine learning versus a deep learning mechanism work. For instance, machine learning algorithms typically divide the learning processes into smaller steps and finally combine the results from each step into one output. Whereas in deep learning, the learning process resolves the problem on an end-to-end basis. For typical machine learning algorithms, external processes identify and create the required features, whereas deep learning algorithms learn high-level features from data and create new features by themselves [103]. Figure 8 shows the differences between the two approaches using an example NLP task. Both can go through some common preprocessing techniques such as tokenization, stemming, and stop-word removal. However, the steps after that are different for traditional machine learning and deep learning. As we have discussed in section 2.3.1, various vectorization techniques are commonly used for machine learning while deep learning techniques leverage word embeddings.

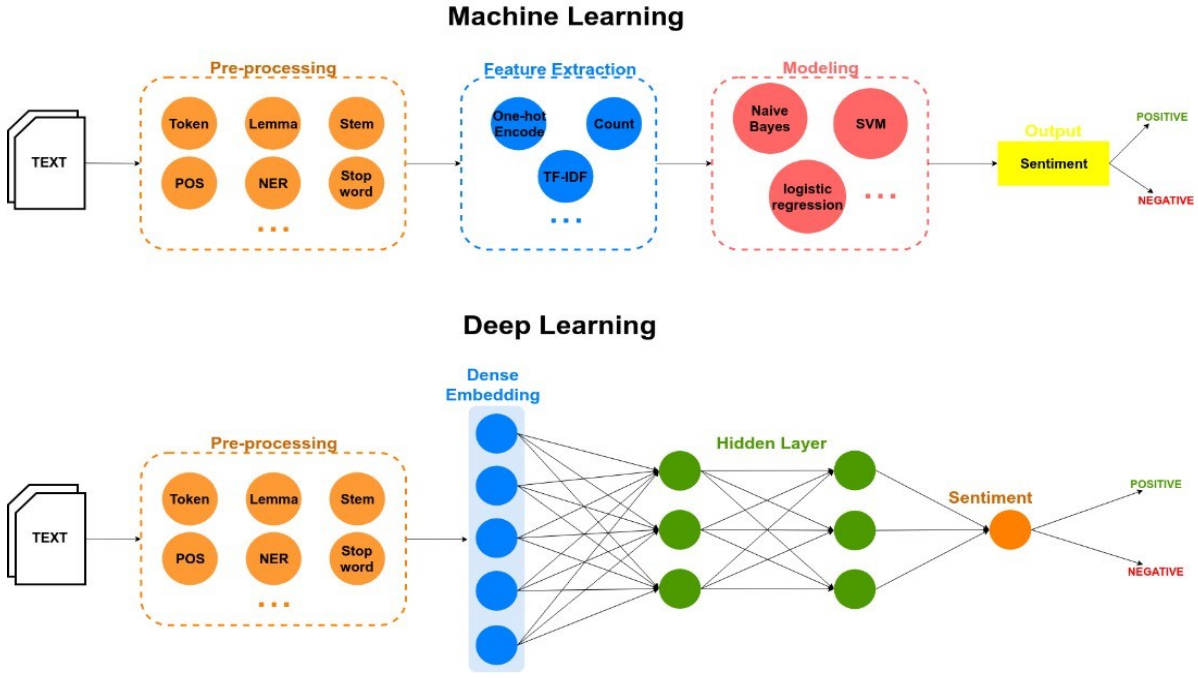

Figure 8: Machine learning and deep learning process [20]

After the data preprocessing is complete, the preprocessed training data is fed into the learning algorithms for training. For traditional machine learning, popular learning algorithms include Logistic Regression (LR) [108], Support Vector Machines (SVMs) [109], Maximum Entropy (MaxEnt) [110] and Naive Bayes (NB) [111]. In the case of deep learning, it is typically a variation

of different Artificial Neural Networks (ANN). There are various types of neural networks such as Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long-Short Term Memory (LSTM) Neural Networks [112]. The neural network attempts to mimic the way the biological neurons in the human brain signal to one another [20]. These networks are made up of layers, containing an input layer, one or more hidden layers, and an output layer (see Figure 9). Each node (i.e., artificial neuron, depicted as h_n in Figure 9) connects to another and has an associated weight and threshold. The node is activated based on the activation function [113]. For instance, it can be activated when the output of any individual node is above the specified threshold value. Then it sends data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. At a high level, in a typical supervised learning setting, a neural network processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the layers, causing the nodes to adjust the weights which control the network. This process occurs over and over as the weights are continually tweaked [114]. At the end of this process, we get the trained model that can be used for various NLP tasks.

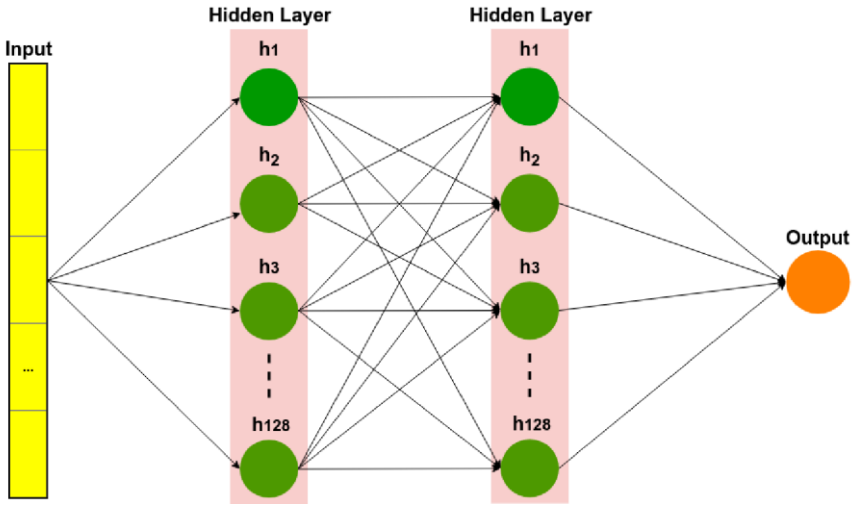

Figure 9: Deep Neural Network [20]

The landscape of deep learning technologies continues to evolve, with the introduction of increasingly sophisticated models occurring on a regular basis. We provide a few relevant examples of some recent deep learning technologies below:

Generative Adversarial Network (GAN): GAN consists of two components: generator and discriminator. During training, the generator uses random noise to create new synthetic data that closely resembles real data [103]. The discriminator uses this output from the generator as input and leverages real data to determine whether the generated data is real or synthetic. Here the generator and the discriminator are competing with each other in an adversarial setting, hence the naming. The output from the discriminator is fed back to the network to improve the respective goals of both components.

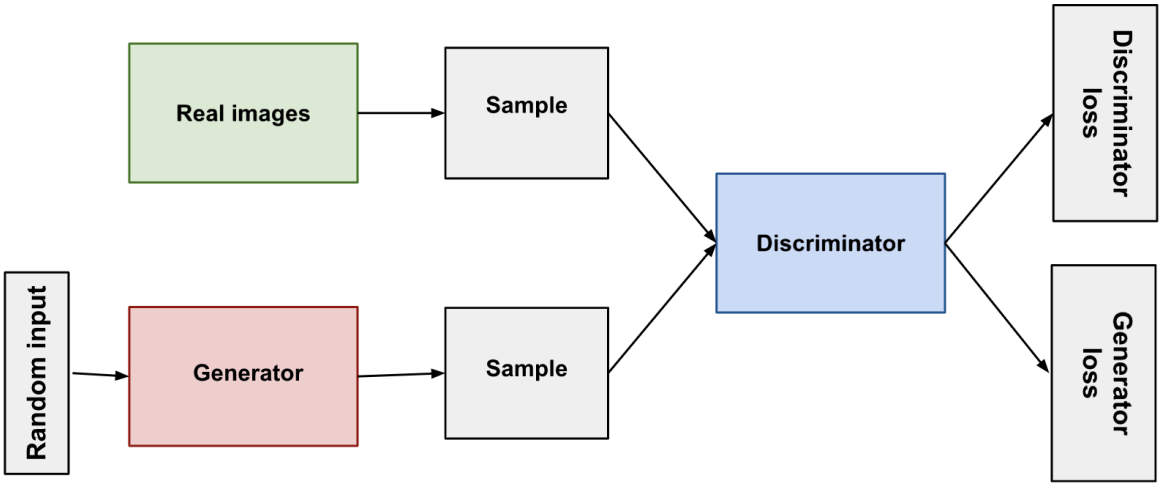

Figure 10: A typical GAN architecture [115]

Transfer learning: Transfer learning aims to store knowledge gained while solving one problem and then apply it to a different but related problem [116]. Deep learning typically requires a large amount of training data, high-end computing resources, and a long training duration. Transfer learning is hence beneficial as it can alleviate all these requirements when used appropriately. The first set of layers in deep neural networks usually contains lower-level features which are more generic, whereas the final set of layers contains higher-level features that are closer to the specific domain or task in question. This final layer can be repurposed to fit a new domain or task. This allows for saving significant resources while training the new model [103].

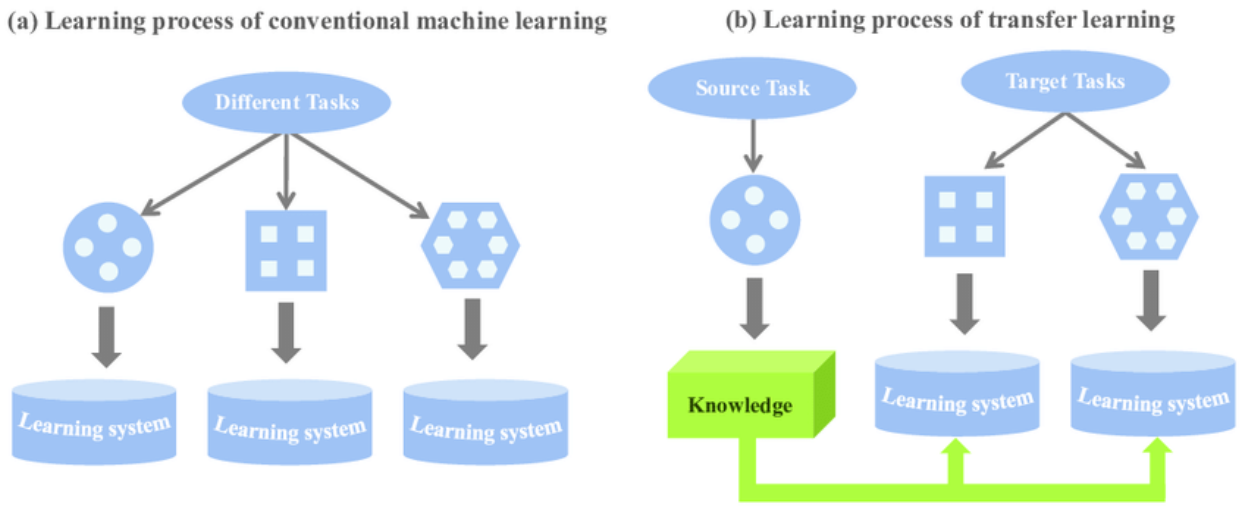

Figure 11: Conventional learning vs transfer learning [117]

For instance, when a model is trained to perform Emotion-Cause Analysis spanning multiple domains of topics such as sports and movies; in a conventional learning process, a new machine learning model is trained on both. In the transfer learning paradigm, if there is an existing model that has been trained on Emotion-Cause Analysis for the sports domain, the new model can leverage the knowledge (see Figure 11) of that model without retraining itself on that domain, and only train on the "movies" domain. Thus, transfer learning can significantly reduce training time and cost.

Transformers: Transformer is an encoder-decoder based neural network that introduced the usage of self-attention to perform tasks involving sequence [95]. Encoders consist of two major components: a self-attention mechanism and a feed-forward neural network. In deep learning, attention refers to attending to different parts of another sequence (i.e., sequence of texts represented as vectors) typically, output sequence and connecting it back to the input sequence [118]. In contrast, self-attention refers to attending to different parts of the same input sequence, facilitating learning about itself [95]. The self-attention mechanism within the encoder takes in a set of input encodings from the previous encoder and weighs their relevance to each other to generate a set of enriched encodings [119]. In simpler terms, self-attention can help understand the relevance of words to other words in a sentence, creating an overall understanding of the likelihood of different words that may appear together. The feed-forward neural network then further processes each enriched encoding individually. On the contrary, each decoder consists of three

major components: a self-attention mechanism, an attention mechanism over the encodings, and a feed-forward neural network. The decoder functions in a similar fashion to the encoder, but an additional attention mechanism is inserted which instead draws relevant information from the encodings generated by the encoders. Figure 12 demonstrates the Transformer architecture.

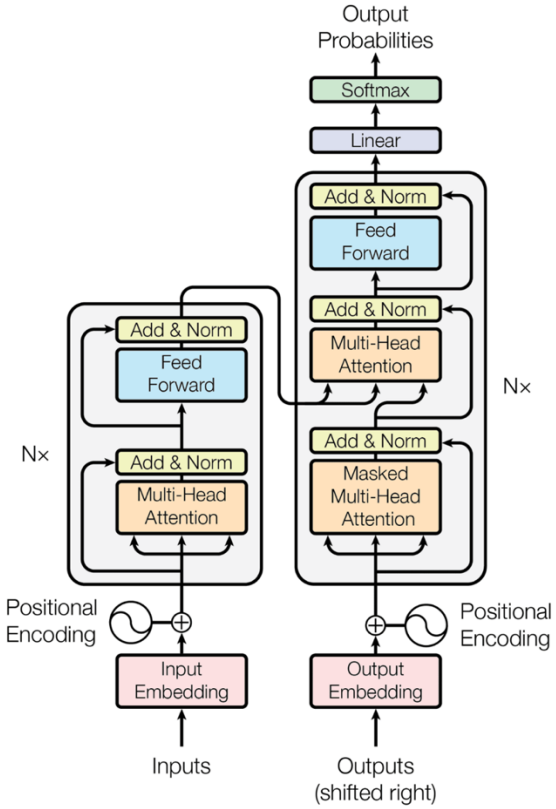

Figure 12: Transformer Architecture [95]

In contrast with any encoder-decoder architecture (which existed before Transformer, i.e., [118]), the main novelty of Transformers is that it introduces self-attention to perform tasks involving sequences, which were typically done by using an RNN-based system with a costly training process [95]. Though the original Transformer used the encoder-decoder combination, there are many models that are based on either the encoder or the decoder but are still referred to as Transformer-based models as they leverage the encoder or decoder in a very similar manner proposed in the original Transformer [95]. For example, Bidirectional Encoder Representations from Transformers (BERT) [120] is an encoder-only Transformer based model while Generative Pre-trained Transformer (GPT) [121] is a decoder-only Transformer-based model.

Language Models: Large-scale pre-trained models (e.g., [122]) have been a key factor in the recent advancements in computer vision (CV). This encouraged the NLP community to adopt the same approach. Starting with the acceleration of convergence speed in NLP models by initializing the word embeddings with pre-trained ones, such as Word2Vec [123] and GloVe [69]. However, these pre-trained word embeddings lack the understanding of the context-dependent nature of languages. To address this, some researchers attempted to use pre-trained RNN models as the backbone for various text classification tasks [124]–[126]. These efforts laid the foundation for the development of large pre-trained language models (LLMs). Building upon these, Devlin et al. [120] and Radford et al. [127] proposed self-supervised pre-training large Transformer models on large text corpora. This demonstrated a significant improvement in the performance of downstream NLP tasks compared to previous state-of-the-art (SOTA) systems. Following this phenomenon, the pre-training and fine-tuning paradigm has become the dominant approach in NLP. Continued innovation in this space has resulted in many popular pretrained language models with increasingly powerful capabilities such as GPT-2 [128], GPT-3 [129], OPT [130], and more recently, ChatGPT [131].

This thesis has a focus on using machine learning technologies with as little training as possible. Consequently, as we shall see in the subsequent chapters, we heavily leverage pre-trained LLMs in our proposed technical solutions for various tasks.

Pretraining and Fine-tuning: Pretraining refers to the process of training a model on a vast set of unlabeled data, usually with an unsupervised learning technique, in order to acquire general features or representations of the input data [132]. Fine-tuning has its roots in the Transfer learning paradigm [116]. Fine-tuning model refers to training it again using a new dataset that is tailored to a specific task after the model has already been pre-trained on a larger dataset. The new dataset includes labeled data, and the aim of fine-tuning is to modify the parameters of the pre-trained model so that it can fit the new data more effectively. The practice of fine-tuning pre-trained language models for downstream tasks gained significant popularity with the emergence of large-scale pre-trained languages models like BERT [120], and GPT [121].

Few-shot, One-shot, Zero-shot learning: Few-shot learning (also known as: low-shot learning) involves training a model on a small number of examples from a new task not known to the model [133]**.** This approach aims to teach the model to learn and generalize from a limited amount of

data, making it more adaptable to new tasks with limited labeled data. In contrast, one-shot learning involves learning from a single example, although sometimes learning from a few examples is interchangeably referred to as one-shot learning [133], [134].

Zero-shot learning refers to the process of training a model to identify new classes that are not included in the training dataset. During training, the model is exposed to a labeled dataset that includes certain classes. Subsequently, the model is tested on a set of classes that it has never encountered before. The expectation is that the model can recognize the new classes based on their attributes or descriptions.

In this chapter, we discussed the machine learning technologies that we leverage throughout our research. We introduced the two main application domains explored in this thesis: Sentiment Analysis and Emotion-Cause Analysis. While in this chapter we summarized the general research trend in these two areas, we discuss the specific gaps in the literature in the subsequent chapters: Chapter 3, 4, 5, and 6. Each of these chapters focus on specific gaps in the literature and the associated research questions that we listed in Chapter 1. Following an integrated article format [2], each chapter provides contextual information to clarify the research questions supported by brief and focused discussions on the literature related to the specific research question. Subsequently, after this introduction to the specific research question, each chapter describes the proposed solutions, associated experiments, discussions, and concluding remarks.

Chapter 3: Multiclass Sentiment Analysis with Limited Labeled Data

Performing Sentiment Analysis with high accuracy using machine learning techniques requires a large quantity of training data. However, such extensive training and access to such a large quantity of labeled data for specific domains can be expensive and time-consuming. This warrants developing more efficient techniques that can perform Sentiment Analysis with high accuracy with a few labeled training data. In this chapter, we aim to address this problem with two proposed solutions, SG-Elect and GAN-BElectra. With rigorous experiments, we investigate the performance of these two architectures in comparison to their respective selected baseline machine learning techniques in terms of performance in multiclass Sentiment Analysis with limited labeled data.

The contributions presented in this chapter have been published in one conference[1] and one journal[2].

3.1 Introduction

In recent times, high-performance computing along with cloud technologies facilitated the rapid growth of many research domains. One such domain is machine learning and applied research based on machine learning techniques such as self-driving cars and language translation services [135]. The dramatic rise in internet usage and social media applications has made most people with connected devices content generators of some sort. These user-generated contents are essential in the growth of many machine learning-based applications such as virtual personal assistants, and recommender systems [136]. The influx of user-generated content elevated the demand for efficient techniques to analyze them to generate valuable insights for government and companies alike. This also caught the attention of researchers who are designing increasingly efficient and accurate data analysis techniques in a frequent manner. Natural Language Processing (NLP) of textual data is one such data analysis domain that saw rapid innovation in recent times.

[1] Riyadh, M., & Shafiq, M. O. (2021). Towards Multi-class Sentiment Analysis with Limited Labeled Data. In 2021 IEEE International Conference on Big Data (Big Data) (pp. 4955-4964).
[2] Riyadh, M., & Shafiq, M. O. (2022). GAN-BElectra: Enhanced Multi-class Sentiment Analysis with Limited Labeled Data. Taylor and Francis Journal of Applied Artificial Intelligence, 36(1), 2083794.

NLP's general goal is to create an understanding of language in computers [137]. Identifying sentiment expressed in a text is a task that contributes to that overall understanding. This task is generally known as "Sentiment Analysis", a research area that has seen rapid progress in recent years [135], [138].

In Sentiment Analysis, a given piece of text is categorized with respect to the sentiment it expresses, often in terms of positive, neutral, or negative. It commonly uses machine learning techniques in order to identify such patterns and then perform these categorizations of textual data. This type of machine learning technique, commonly referred to as supervised learning, heavily relies on training data. Sentiment Analysis is a domain-specific task. This means words used in one context to express certain sentiments may exert different sentiments in different contexts. As a result, training a machine learning model for Sentiment Analysis tasks typically requires training with domain-specific labeled data. Having a large quantity of labeled training data for specific domains can be tedious and expensive. In this research, we focus on the problem of the limited labeled training data in Sentiment Analysis.

Machine learning-based Sentiment Analysis models trained with a low number of training data can perform poorly on this text classification task. However, since a large number of training data may not be always available, researchers have attempted to develop techniques to have higher Sentiment Classification accuracy with a lower number of labeled training data. These techniques, however, still fall short of reaching the peak performance of machine learning models trained on large, labeled datasets. As a result, achieving higher accuracy in Sentiment Classification tasks with a low amount of training data is still a worthwhile research problem to address.

In this chapter, we propose two successive technical solutions, named SG-Elect, and GAN-BElectra to achieve high accuracy in Sentiment Analysis with limited labeled data (i.e., 50 datapoints per class). Both solutions focus on multi-class Sentiment Analysis. In this type of Sentiment Analysis, there are more than two sentiment classes to choose from. The typical class labels are positive, neutral, and negative. This is in contrast with binary classification where only two sentiment classes are used for categorization: positive and negative. Multi-class Sentiment Analysis, especially the ones that consider neutral as one of the classes, seems to possess more practical value compared to binary classification since not every text expresses a sentiment, and some texts naturally fall into the neutral category.

We first develop SG-Elect and evaluate its performance against its baseline technique GAN-BERT [139]. Building upon the learning from this work, we develop GAN-BElectra, for which we use its predecessor, SG-Elect, as the baseline.

In the subsequent sections, we describe the architectures of both techniques in detail. After that, we discuss the experiments and evaluations related to both techniques in concert to facilitate a convenient understanding and comparison of both techniques. This is also to account for the overlaps between the two studies incurred by the fact that GAN-BElectra builds upon SG-Elect.

3.2 Related Studies

The lack of labeled data is a common problem in text classification tasks. There are several proposals in the literature to address this general problem [139]–[141]. Using lexicons to classify texts is a typical solution offered for this issue. [142]. However, the lack of accuracy of lexicon-based techniques in text classification, especially in comparison to fully supervised machine learning-based methods is well known [143].

Semi-supervised learning is another common technique that attempts to resolve the issue of the lack of labeled training data. This method is specially engineered to train machine learning models with a few labeled data [144], [145]. Researchers have designed several variations of semi-supervised learning. These include using the teacher-student method where teacher confidence is utilized to identify easy samples during training (e.g., self-paced co-training [146], self-paced learning [144]). Other examples include utilizing meta-learning [147] and active learning [145] for sample selection based on teacher confidence. A recent application of semi-supervised learning, specifically for Sentiment Analysis, is a neural network-based semi-supervised learning framework proposed by Li et al. [107]. It performs training with a few labeled data along with many unlabeled data. To address the labeled data scarcity issue in short text classification tasks such as Sentiment Analysis of tweets, Yang et al. [148] proposed a heterogeneous graph attention network that embedded a flexible heterogeneous information network framework that modeled short text with the functionality to include additional information while appropriately detecting their semantic relations. By extending their technique with semi-supervised inductive learning, they demonstrate that the proposed solution outperforms their selected SOTA baselines for both single and multi-label classification tasks. Kim et al. [149] proposed a novel self-training method

that leveraged a lexicon to guide its mechanism in generating pseudo-labels in order to address the lack of labeled data in text classification, particularly Sentiment Analysis. They demonstrated that the guidance from the lexicon in their experimental setup enhanced the reliability of pseudo labels by performing manipulation on the loss term.

Abonizio et al. [150] conducted an in-depth study of the usage of text augmentation (i.e., creation of synthetic textual data by modifying existing text with the goal of increasing the diversity and size of training data for NLP tasks) in addressing labeled data scarcity issues in Sentiment Analysis. They offered a taxonomy for these techniques, first categorizing all techniques into "sentence" manipulation and "embedding" manipulation (i.e., manipulating representative vectors of text instead of the actual text data), and then further dividing the "sentence" manipulation category into three subcategories: transformation (e.g., synonym replacement), paraphrasing (e.g., adapting translation models to generate rephrased text in the same language), and generation (e.g., using autoregressive language models such as GPT2 [128] to generate text). They evaluated various text augmentation techniques and observed how they influenced the Sentiment Classification accuracy of different techniques in scenarios such as a low number of labeled training data. For instance, they found that BERT [120] and ERNIE [151] achieved superior classification performance with a low number of available training samples when boosted with the back-translation [152] augmentation technique. Some researchers also applied text augmentation techniques to mitigate the labeled data scarcity in non-English languages. For example, Barriere et al. [153] used automatic translation of English tweets to French, Spanish, German, and Italian to apply data augmentation which improved the Sentiment Analysis performance over non-English tweets using different transformer-based techniques [119]. Edwards et al. [154] demonstrated how leveraging GPT-2 [128] driven text augmentation in a few-shot learning setup could enhance text classification accuracy.

Recent advances in the pre-trained language model made their landfall in the Sentiment Analysis research area inevitable. Examples of such pre-trained models include BERT [120], Electra [155] etc. These models, which are typically based on Transformers [95], are trained on a vast amount of data, typically following an unsupervised or self-supervised approach. This provides these models with a general capability to comprehend language. At this stage, these pre-trained models are capable of many tasks that require an understanding of general-purpose language representation. For leveraging them in downstream tasks that require domain-specific knowledge,

for example, Sentiment Analysis, they go through another lightweight training process known as fine-tuning. This is essentially training the pretrained model with the task/domain-specific labeled data. Since the pretrained model already has an understanding of language in general, fine-tuning on a very small set of domain-specific labeled data can make them significantly more accurate as classifiers compared to typical training of machine learning models. As a result, there has been some interest in the researcher community in using these pretrained models to address the issue labeled data scarcity in text-classification tasks in general. For example, Croce et al. [139] experimented with using a semi-supervised generative adversarial network (GAN) to improve BERT's fine-tuning stage. This architecture, named GAN-BERT [139], was mainly focused on achieving higher accuracy compared to the original BERT network in text classification tasks with a low amount of labeled training data.

Researchers have proposed various approaches that attained decent accuracy in Sentiment Classification tasks with limited labeled data. While approaches that rely on a lot of training data can achieve considerably higher accuracy in this task, achieving similar accuracy with limited labeled data remains a challenge. As a result, achieving higher accuracy in Sentiment Classification tasks using a few labeled data remains a worthwhile investigation.

In the subsequent sections, we first introduce the two successive techniques we developed to perform multi-class Sentiment Analysis with limited labeled data (i.e., 50 datapoints per class), named SG-Elect, and GAN-BElectra. We begin with outlining the contributions of each technique, followed by in-depth discussions on the proposed architectures.

3.3 SG-Elect

3.3.1 Contributions

SG-Elect outperforms a recently published technique (i.e., GAN-BERT [139]) in multiclass Sentiment Classification tasks with limited labeled data in terms of various classification metrics (i.e., F1 macro and F1 weighted average scores). We employ cutting-edge deep learning technologies such as transformer-based pre-trained models together with more traditional machine learning algorithms to devise our solution. With SG-Elect, we make the following contributions:

- We propose a multi-class Sentiment Analysis technique named SG-Elect that improves upon our SOTA baseline GAN-BERT in terms of sentiment classification performance (F2 macro, F1 weighted average score) with limited labeled training data.
- We evaluate our proposed technique with three publicly available datasets to demonstrate the efficacy of our technique in terms of classification accuracy compared to our SOTA baseline [139].
- We evaluate the performance of individual components used in SG-Elect for an in-depth understanding of its architecture.

3.3.2 Architecture

As discussed in our literature review, the usage of data with pseudo labels is a well-established technique to mitigate the effect of limited labeled data. Drawing upon the principles of leveraging data with pseudo labels and the paradigm of semi-supervised learning in limited labeled data setting [144], [145], we propose SG-Elect, a Sentiment Analysis technique for multi-class Sentiment Analysis with limited number of labeled training data, which consists of two primary components: a pseudo-label generator and two-step fine-tuning of the final classifier. We discuss this architecture in detail below.

3.3.2.1 Pseudo label generator

This part of the architecture is responsible for generating pseudo labels based on the few original labeled data and a large amount of unlabeled data. This component combines techniques from both the traditional machine learning and deep learning paradigms. It includes the following subcomponents:

3.3.2.1.1 SS-Trainer

SS-Trainer (i.e., Semi-Supervised Self Trainer) subcomponent leverages the Self-Training mechanism [156], [157], which allows the supervised classifier to function as a semi-supervised classifier, which iteratively produces pseudo labels and include them in the training data. The iteration continues until a maximum number of iterations is reached, which we selected as 1000 based on our experiments. For the probability threshold criterion, we used 0.80, meaning, the algorithm will only add a pseudo label to the training set if the probability meets this threshold. As the base estimator of the Self Training Classifier, we used a stacked classifier (i.e., ensemble). This

stacked classifier consists of an SGD Classifier (i.e., machine learning algorithm that uses the SGD or Stochastic Gradient Descent optimization algorithm to train a linear classifier) [158] and an XGB Classifier [159] (Figure 13).

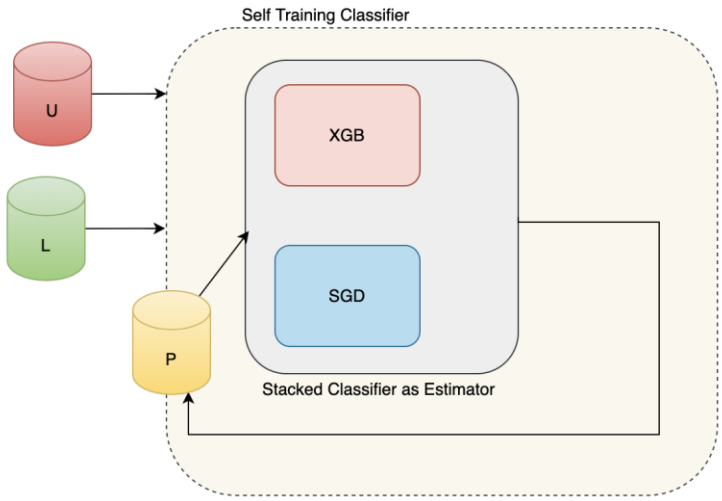

Figure 13: Semi-supervised stacked classifier subcomponent

SGD Classifier employs an efficient method to perform classification tasks such as Sentiment Classification. We use logarithmic loss which produces logistic regression, eventually making it a probabilistic classifier. For the regularization term (i.e., penalty), we use L2, which is a standard regularizer for this linear classifier. To multiply the regularization term, we use a constant value (i.e., named alpha) of 0.00001, which is used within the algorithm to automatically determine the optimal learning rate. The maximum number of passing over the training data (i.e., iterations) is 1000, while the tolerance is 0.001, both are standard choices that have been shown to provide the best outcome in our experiments.

XGB Classifier comes from the family of XGBoost which is an optimized distributed boosting mechanism often used to improve the efficiency of learning algorithms [159]. In our experimental setting, the combination of the XGB Classifier with its standard configuration with the SGD Classifier provided the best outcome in terms of pseudo-label generation compared to some other combinations involving classifiers of SVM and NB variants. Figure 13 represents this Self Training Classifier subcomponent where it begins training with the labeled (L) and unlabeled data (U) and, the stack classifier incrementally generates some pseudo labels which are then fed back

(data with pseudo labels, P) to the model as training data. This process is continued until all the data that is unlabeled gets a pseudo-label counterpart.

3.3.2.1.2 GAN-BERT

GAN-BERT is a deep learning network that contains BERT (Bidirectional Encoder Representations from Transformers) [146], and SS-GAN (Semi-supervised Generative Adversarial Network) (Croce, Castellucci, and Basili 2020). BERT is a pretrained model based on Transformer technology [95]. It learns contextual relations between words using an attention-based mechanism. The model was pretrained with large training data consisting of raw texts. The pre-trained model is then fine-tuned on training data for a specific task.

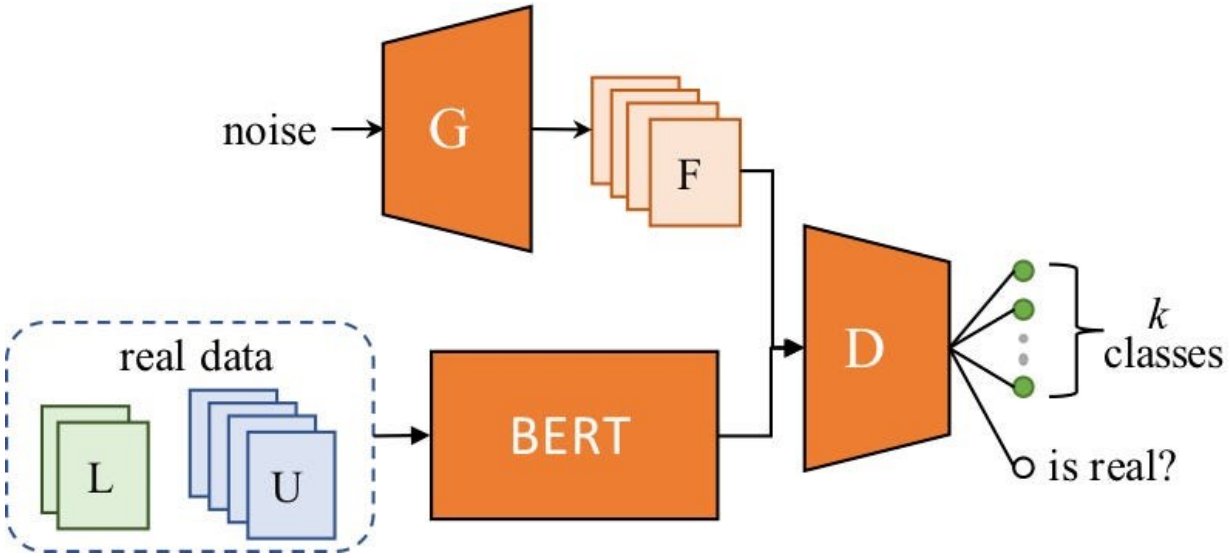

Figure 14: GAN-BERT Architecture [139]. U, L, F, G, D denote unlabeled data, labeled data, fake labels, generator, and discriminator respectively

GAN-BERT's novelty mainly stems from the fact that it extends BERT's fine-tuning phase with an SS-GAN. SS-GAN consists of two main parts: a) a Generator and b) a Discriminator as shown in Figure 14. The Generator generates synthetic labels, and the Discriminator classifies those labels into real and fake using adversarial learning. In GAN-BERT, a pre-trained BERT model is fine-tuned with task-specific layers first, which is part of BERT's typical fine-tuning process. Second, during this fine-tuning stage, SS-GAN layers are used to enable semi-supervised learning.

We use GAN-BERT in its original configuration from the study [139]. At first, we use GAN-BERT's fine-tuning process to fine-tune the model with the original few labeled data. Next, the fine-tuned GAN-BERT model is used to generate pseudo labels for the unlabeled data.

3.3.2.1.3 Combinator

The typical outputs of both Semi-Supervised Self Training Classifier (i.e., traditional machine learning) and GAN-BERT (i.e., deep learning) are predicted labels. Along with the predicted labels, the next subcomponent in our architecture requires the probability of each label in order to choose the best pseudo label. As a result, we also derive the probability for each prediction for each classifier and pass it to the next subcomponent "Combinator" (see Figure 16). Combinator uses this probability score to select the best pseudo label from the pseudo labels generated by the two pseudo label generators for each unlabeled datapoints. For example, for a given sentence, if GAN-BERT's predicted pseudo label is "Negative" with a probability score of 0.75 and SS-Trainer's predicted pseudo label is "Neutral" with a probability score of 0.65, then the combinator will choose GAN-BERT's predicted pseudo label "Negative" since it has a higher probability score. This selected pseudo label along with the associated text then becomes the input for the next component within the architecture which we discuss below.

3.3.2.2 Electra

This final component involves a deep learning model (Figure 15). This model includes multiple deep-learning layers. Immediately after the input layer, we use the Transformer based pre-trained model which is known as Electra (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) [155].

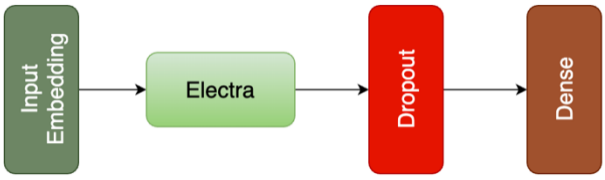

Figure 15: Electra-based pretrained component

Electra implements a more efficient training mechanism compared to the BERT-like masked language models (MLM). BERT-like MLMs utilize subsets of unlabeled input in the pre-training stage, where the network learns to identify the masked tokens and then redeem the original input.

In Electra, the improvement is primarily achieved through corrupting the input with replaced tokens instead of using a masked subset of input and then pre-training the network as a discriminator that identifies original tokens vs. replaced ones. In our architecture, the "pooled output" from this Electra layer goes to a Dropout Layer which prevents the model from overfitting. The next layer is a deeply connected neural network layer (i.e., Dense layer). The output of this layer is passed to an Argmax function in order to calculate the final predicted labels, as shown in Figure 15.

This final classifier model involves AdamW [160] as the optimizer, which is a stochastic optimization method that carries out modifications in the typical implementation of weight decay in Adam optimizer [161]. It achieves this by separating weight decay from the gradient update. For calculating loss, we use Sparse Categorical Cross-entropy since the classification task we are performing involves more than two mutually exclusive classes. We observed this classifier to reach the peak performance at 10 epochs, which is what we set as the number of epochs. The classifier is fine-tuned twice, once with the data with pseudo labels generated in the previous component, and the next using the original few labeled data.

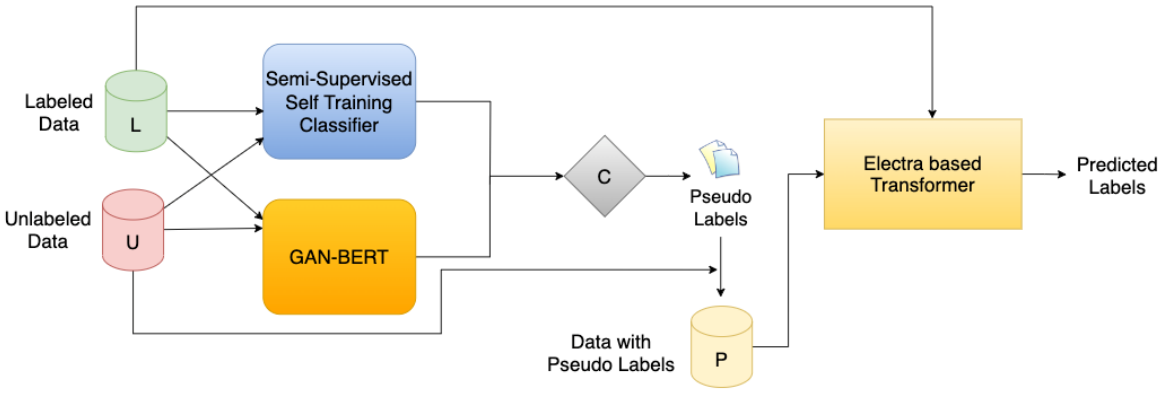

Figure 16: SG-Elect Architecture. U, L, P denote unlabeled, labeled, and pseudo-labeled data respectively

Algorithm 1: Overall Training Process of SG-Elect. Meanings of symbols used: C_{SS}: SS-Trainer, C_{GB}: GAN-BERT, C_{EL}: Electra, Y_{SSL}: Predicted label by C_{SS}, Y_{SSC}: Probability score of each prediction by C_{SS}, Y_{SS}: Collection of all predicted labels and associated probability scores by C_{SS}, Y_{GBL}: Predicted label by C_{GB}, Y_{GBC}: Probability score of each prediction by C_{GB}, Y_{GB}: Collection of all predicted labels and associated probability scores by C_{GB}, Y_P: Collection of final selected pseudo labels, X_P: Collection of data with pseudo labels

```
inputs: X = X_L + X_U
X_L = x_L1, x_L2, …, x_Ln
X_U= x_U1, x_U2, …, x_Um
outputs: Trained model
BEGIN
      for all x ∈ X do {
            C_SS ← Train (SGD + XGB, x)
      } end for
      return C_SS
      for all x_U ∈ X_U do {
            Y_SSL, Y_SSC ← C_SS
      } end for
      return Y_SS
      for all x ∈ X do {
            C_GB ← Train (GAN-BERT, x)
      } end for
      return C_GB
      for all x_U ∈ X_U do {
            Y_GBL, Y_GBC ← C_GB
      } end for
      return Y_GB
      for all x_U ∈ X_U do {
            Y_P
            if Y_GBC > Y_SSC
                  add Y_GBL to Y_P
            else
                  add Y_SSL to Y_P
      } end for
      return Y_P
      Init X_P //to store data with pseudo labels
      for all x_U ∈ X_U do {
            Append (x_U,y_P) to X_P
      } end for
      for all x_P ∈ X_P do {
            C_EL ← Train (ElectraTransformer, x_P)
      } end for
      return C_EL
      for all x_L ∈ X_L do {
            C_EL ← Train (ElectraTransformer, x_L)
      } end for
      return C_EL
END
```

Figure 16 shows the overall architecture of our solution.

Algorithm 1 captures its overall training process. The algorithm for training our solution includes the following primary steps:

- We start with a few labeled (X_L) and many unlabeled data (X_U).
- Using this input data for the training process, the Semi-supervised Self Training Classifier (C_{SS}) and GAN-BERT component (C_{GB}) generate pseudo labels (Y_{SS}, Y_{GB} respectively) for the unlabeled data.
- Based on the confidence of prediction, the next subcomponent, 'Combinator' selects the best predictions (Y_P) for each datapoint. We form dataset X_P using these pseudo labels along with the associated text from X_U, to fine-tune an Electra Transformer-based pre-trained model (C_{EL}).
- Finally, another fine-tuning process is done by using the original few labeled data.
- In the end, we run our test data on the fine-tuned Electra Transformer which generates the final predicted labels for our architecture.

3.4 GAN-BElectra

3.4.1 Contributions

GAN-BElectra builds upon the technique discussed in the previous section, SG-Elect, [162] with the aim to reduce architecture complexity and training steps without sacrificing performance gain achieved by its predecessor. SG-Elect employs three machine learning components. Two of them are deep learning components (GAN-BERT [139] and Electra [155]) and the other one is a traditional machine learning component (Semi-Supervised Self Trainer [156]. In GAN-BElectra, which is an evolution of SG-Elect, we employ only two deep learning components (GAN-BERT and Electra), which significantly reduces the complexity of the architecture and training steps, while still achieving a small performance gain (arithmetically) in Sentiment Classification over SG-Elect.

As part of devising GAN-BElectra, we make the following contributions:

- We propose a novel technique for multi-class Sentiment Analysis named GAN-BElectra which builds upon SG-Elect. It possesses a significantly more lightweight architecture and requires less training step compared to SG-Elect, without sacrificing any performance gain

achieved by its predecessor in multi-class Sentiment Analysis with limited labeled training data (i.e., 50 labeled datapoints per class).

- We evaluate GAN-BElectra on three datasets that are available in the public domain. These experiments illustrate that GAN-BElectra outperforms its state-of-the-art (SOTA) baseline (i.e., SG-Elect) in the multi-class Sentiment Analysis task with limited labeled training data.
- We dissect GAN-BElectra's architecture and analyze the individual components that constitute it to facilitate a thorough understanding of the proposed solution.

3.4.2 Architecture

GAN-BElectra consists of two primary components: GAN-BERT [139] and Electra [155]. These two components have been described above in greater detail in the SG-Elect's architecture and they are leveraged in GAN-BElectra with similar configuration and use-case.

GAN-BElectra's architectural design is built upon the same principles that guided the design of SG-Elect's architecture: the utility of data with pseudo labels and the semi-supervised learning methods in mitigating the effect of limited labeled training data in machine learning models [144], [145]. In GAN-BElectra, GAN-BERT acts as the sole pseudo label generator whereas Electra consumes the data with those pseudo labels to fine-tune itself.

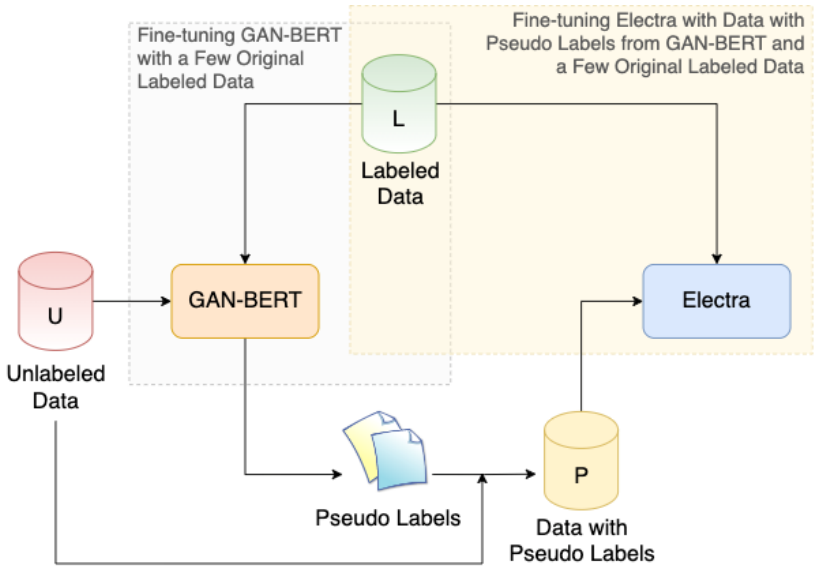

Figure 17: GAN-BElectra architecture. U, L, P denote unlabeled, labeled, and pseudo-labeled data respectively

Figure 17 shows the overall architecture of GAN-BElectra. This architecture evolved from SG-Elect where there was an additional Semi-Supervised Self Training Classifier (SS-Trainer) that acted as another pseudo-label generator in addition to GAN-BERT.

GAN-BElectra streamlines SG-Elect architecture by stripping away the process-heavy SS-Trainer, leaving only GAN-BERT as the sole generator of pseudo labels. This also abolishes the need of having the “Combinator” component which further simplifies the architecture.

Algorithm 2: Overall training process of GAN-BElectra. Meanings of symbols used: C_{GB}: GAN-BERT, C_{EL}: Electra, Y_{GB}: Collection of pseudo labels generated by C_{GB}, X_P: Collection of data with pseudo labels

```
inputs: X = X_L + X_U
X_L = x_L1, x_L2, …, x_Ln
X_U= x_U1, x_U2, …, x_Um
outputs: Trained model
BEGIN
  for all x ∈ X do {
      C_GB ← Train (GAN-BERT, X)
  } end for
  return C_GB
  for all x_U ∈ X_U do {
      Y_GB ← Predict (C_GB, X_U)
  } end for
  return Y_GB
  Init X_P //to store data with pseudo labels
  for all x_U ∈ X_U do {
     Append (x_U,y_GB) to X_P
  } end for
  for all x_P ∈ X_P do {
      C_EL ← Train (ElectraTransformer, x_P)
  } end for
  return C_EL
  for all x_L ∈ X_L do {
      C_EL ← Train (ElectraTransformer, X_L)
  } end for
  return C_EL
END
```

Algorithm 2 demonstrates GAN-BElectra's high-level training process. The training process begins with a large amount of unlabeled data (X_U) and a few labeled data (X_L). After being trained on the original few labeled data (X_L) in a semi-supervised manner, the GAN-BERT component (C_{GB}) generates pseudo labels (Y_{GB}) for the unlabeled data. We form dataset X_P using these pseudo labels along with the associated text from X_U to fine-tune our Electra-based pretrained model (C_{EL}). This final Electra-based classifier goes through another fine-tuning using our few labeled data. At this point, GAN-BElectra is fully trained, and we evaluate it using our test data.

3.5 Experiments and Evaluation

3.5.1 Procedure

We perform several experiments in order to evaluate GAN-BElectra, SG-Elect (GAN-BElectra's baseline), GAN-BERT (SG-Elect's baseline), and two other constituting techniques, Electra and SS-Trainer. Although individual experiments have some specificities, they all share some common steps. These include:

3.5.1.1 Data splitting

One of the initial steps in our experiments involves splitting the dataset into training and test sets. For the train-test split, we opted for an 80:20 ratio where 80% of the total data accounted for the training dataset and 20% is allocated for testing. We also split the training dataset into labeled and unlabeled sets. In all our data-splitting exercises, we use specific seed numbers to generate controlled randomness. This makes our experiments reproducible. It is noteworthy that during the development phase, particularly for hyperparameter tuning, to mitigate the possibility of overfitting to the test data from each dataset, we leveraged random samples from one of our selected datasets. These selected hyperparameters were later used in all out experiments across all datasets.

Once we split our training dataset between labeled and unlabeled sets, we remove the labels from the unlabeled dataset to serve its intended purpose. The labeled dataset contains 50 datapoints for each sentiment class, and the remaining training dataset contains unlabeled data. With only 50 datapoints per class, the labeled dataset represents a minute fraction of the total training datapoints used across our experiments with various datasets.

Dataset	Percentage of Labeled data in Training Data
SST5	2.91%
US Airline	1.28%
SemEval	0.31%

Table 2: Percentage of labeled data in total training data

3.5.1.2 Data preprocessing

After training (labeled, and unlabeled) and test datasets are created, we perform additional data pre-processing such as the removal of stop words as necessary, and performing vectorization of the datasets so that they are consumable by the machine learning algorithms. We describe this in greater detail in the section that follows.

3.5.1.3 Defining model

The next step is to define the machine learning model. This involves defining the correct parameters and configuration for the model, in addition to defining the layers that eventually construct the deep learning networks used in this study.

3.5.1.4 Training and testing

After the model definition step, we begin training the model with our training data which is different for each experiment. Following the completion of the training, we test our trained model with the test datasets. We report various metrics including F1-score, and accuracy based on our evaluations of the models.

To sum up, each experiment mainly consists of dataset splitting, data preprocessing, defining the model, training the model, and evaluating the trained model. To make our experimental findings more reliable, we run each of our experiments three times using three different random seeds, and we report the average results from these experiments. It is noteworthy that since GAN-BElectra builds upon SG-Elect [162] and the evaluation criteria, datasets, and experimental settings are intentionally identical in these two consecutive studies and they share some of the reported results, especially as it relates to the two common individual sub-components, namely GAN-BERT [139] and Electra [155]. Consequently, we report results from both studies in a consolidated manner.

3.5.2 Tools

Our experiments required leveraging several existing tools. The following are some noteworthy examples:

3.5.2.1 Software:

Programming language: We use Python in all our experiments. Python provides many useful libraries for NLP tasks out of the box as well as for machine learning experiments in general.

Libraries: We use many publicly available software libraries in our experiments as required. Some of the noteworthy ones include Numpy [163] and Matplotlib [164]. For deep learning algorithms, we heavily leverage TensorFlow [165].

Machine learning models: We use GAN-BERT in its original configuration [166]. This serves as the common pseudo-label generator in both GAN-BElectra and SG-Elect. For SG-Elect, we also use SS-Trainer in addition to GAN-BERT for pseudo-label generation. As the final classifier, both solutions leverage the large variant of the pre-trained Electra model [167].

3.5.2.2 Hardware

Instead of using our on-premise hardware resources, we utilized cloud hardware resources provided as part of the Pro-tier subscription of Google Colab [168]. This platform provides a browser-based user interface to perform coding, which is specially designed for data science tasks. This is supported by backend resources such as Tensor Processing Unit (TPU) [169] and Graphics Processing Unit (GPU) [170]. The hardware specification provided by Google Colab Pro varied for different experiments and the runtime engines are dynamically allocated. However, the following specification represents the typical maximum hardware resource we received from Google Colab Pro:

- Compute: Intel(R) Xeon(R) CPU @ 2.30GHz (Max. available cores: 40)
- Memory: 36 GB
- Disk: 226 GB

3.5.3 Datasets

In order to perform a robust evaluation of our solution, we chose these three datasets: SST5 dataset [171], US Airline dataset [172], and SemEval 2017 dataset [173] (we refer to this as *SemEval* in

this thesis). The reasons for choosing these three datasets include, they are all multi-class Sentiment Analysis datasets, publicly available for use for research, and commonly used in Sentiment Analysis research. We describe these datasets below:

3.5.3.1 SST5

The SST5 is a 5-class Sentiment Analysis dataset. SST stands for Stanford Sentiment Treebank [171]. Datasets with 5 or more sentiment classes are typically called fine-grained sentiment datasets. The SST5 dataset contains short texts, and each short text is labeled with any of the following 5 sentiment classes:

- Very Positive
- Positive
- Neutral
- Negative
- Very Negative

Figure 18 demonstrates the composition of this dataset.

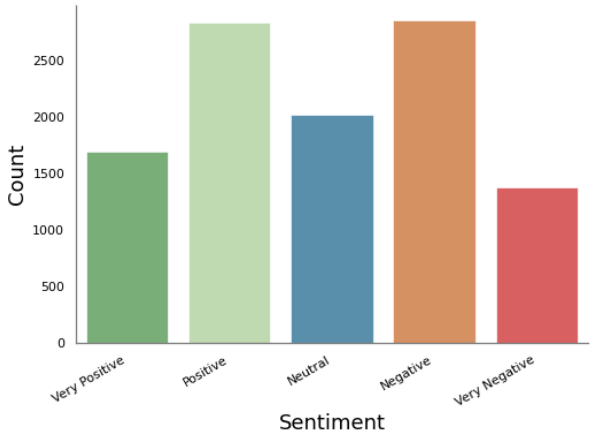

Figure 18: SST5 dataset composition

3.5.3.2 US Airline

The US Airline dataset [172] is a 3-class Sentiment Analysis dataset. Each datapoint consists of short text which is a real Twitter post about US Airline carriers, and a sentiment label which is either Positive, Neutral, or Negative. Figure 19 shows the composition of this dataset.

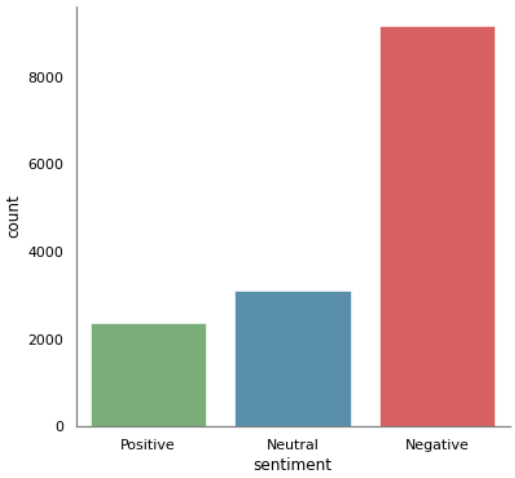

Figure 19: US Airline dataset composition

3.5.3.3 SemEval

SemEval [173] is another 3-class Sentiment Analysis dataset we have used in our experiments. This dataset is composed of short text from real Twitter posts and each short text is labeled as either Positive, Neutral, or Negative. Figure 20 shows the composition of this dataset.

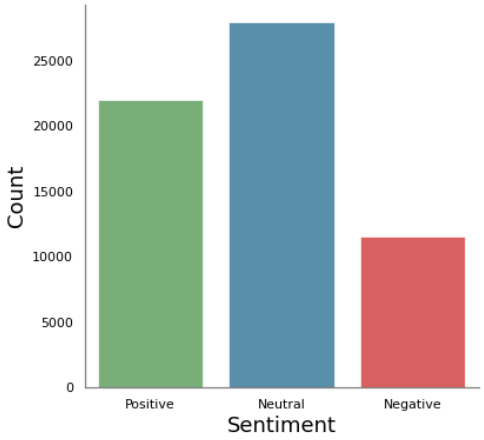

Figure 20: SemEval dataset composition

3.5.4 Data preprocessing

We use the BERT tokenizer [165] to transform our input texts into a numerical format. The deep learning models incorporated in our experiments anticipate all the input sentences concatenated to one another. The input begins with a “[CLS]” token (this indicates that the task is a "classification problem"). The end of each input is indicated with a separator token “[SEP]”. Finally, the

datapoints were represented as Tensors [165], and the deep learning experiments were executed on TPUs [169]

It is noteworthy that since our usage of 50 datapoints per class as the original few labeled data makes the composition of the primary training data organically balanced, we kept the datasets in their original composition without applying any additional balancing technique. Also of note is that the original SemEval dataset contains a total of 61473 datapoints, the majority of which would have been used in our setup as unlabeled data. Since our original labeled data per class is only 50 datapoints, an extensively large amount of unlabeled data provides little value in our setup while significantly increasing the training time. Consequently, we opted to use a representative sample from this dataset (i.e., taking a total of datapoints 20000 out of 61473, resembling the other 3-class dataset used in our study: US Airline dataset) while preserving the original composition (i.e., class distribution) of the dataset by leveraging stratified random sampling [174].

3.5.5 Evaluated Techniques

Using the three selected datasets, we evaluated our proposed techniques: GAN-BElectra, and SG-Elect (GAN-BElectra's baseline). We also evaluate GAN-BERT (SG-Elect's baseline) using the same datasets. In addition, we inspect how other constituting components (Electra, SS-Trainer) of our proposed solutions perform on the same datasets.

3.6 Results

In order to make our findings more reliable, we evaluated each technique (GAN-BElectra, SG-Elect, GAN-BERT, Electra, SS-Trainer) three times on each of the three selected datasets using seed numbers to achieve controlled randomized datapoint allocation for train and test split for each evaluation (for an example of a similar study with three runs per model, see [175]). In summary, we ran a total of 45 experiments representing 9 experiments per technique. We report the average results from the three randomized runs for each evaluation.

Table 3 and Table 4 provide a summary of all our experimental results. In Table 3, we report the mean accuracy (i.e., an average of three runs. Accuracy is calculated using true positives and true negatives as shown below) for each technique on each dataset.

$$accuracy = \frac{tp + tn}{tp + fp + tn + fn}$$

where, tp, fp, tn, fn consecutively represent true positive, false positive, true negative, and false negative.

In addition, we report the standard deviation of this mean, which represents the variations of the accuracy scores across the three evaluations for each technique on each dataset. Table 3 shows that the standard deviation scores for all our experiments were between 0.0111 and 0.0514 indicating high reliability of the reported mean accuracy scores based on our randomized experiments.

Table 3 also contains F1 macro (defined below) which, in contrast with accuracy, provides a more reliable assessment of techniques in the presence of class imbalance [176].

$$F1\ Score = 2 * \frac{pr * re}{pr + re}$$

where,

$$pr\ (precison) = \frac{tp}{tp + fp}$$

$$re\ (recall) = \frac{tp}{tp + fn}$$

$$F1\ (macro) = \frac{\sum_{i=1}^{N} F1\ Score_i}{N}$$

where,

$$N = number\ of\ classes\ in\ the\ dataset$$

In addition to accuracy and F1 macro average, we additionally report F1 weighted average in Table 3, which is typically used when more importance is given to classes with larger support [176].

$$F1\ (weighted) = \sum_{i=1}^{N} w_i\ 2 * F1\ Score_i$$

where,

$$w_i = \frac{No.of\ samples\ in\ class\ i}{Total\ number\ of\ samples}$$

$$N = number\ of\ classes\ in\ the\ dataset$$

We report the mean standard error (i.e., the inverse of accuracy) in Table 4. Inspired by Tom Mitchell's suggestion on comparing machine-learning models [177], we additionally report estimation statistics based on the reported standard error at the commonly used significance level of 95%. The following formula was used for the confidence interval (ci) calculation:

$$ci = z \times \sqrt{\frac{e \times (1 - e)}{n}}$$

where,

- ***z** is a critical value from the Gaussian distribution which has a value of 1.96 for 95% significance level* [178],
- ***n** is the size of the test sample,*
- ***e** is the standard error reported in Table 4.*

The equations reported in this section can be found in [178], [179].

Our analysis suggests that the confidence interval (half width) for the reported standard error remained between the range of 0.0154 and 0.0203 (see Table 4). This arguably further indicates the robustness of our results across multiple experimental runs for each of the evaluated techniques for the selected datasets. We also conduct statistical significance testing on our overall findings on the models' classification performance based on accuracy, F1 macro, and F1 weighted average scores (see Table 11 and Table 12).

Table 6 contains the accuracy score of pseudo-label generation by GAN-BERT and SS-Trainer (only used in SG-Elect). Table 8, Table 9, and Table 10 contain detailed results for all evaluated techniques for SST5, US Airline, and SemEval datasets respectively. Following the tables, we also include confusion matrices visualizing the classification accuracy and true positives and negatives for each evaluated technique for the three selected datasets.

We describe our results for each dataset in the following sections.

Dataset	Technique	Mean F1 Macro	Mean F1 Weighted Avg.	Mean Accuracy	Standard Deviation of Mean Accuracy
SST5	GAN-BElectra	**0.3796**	**0.3764**	**0.3825**	0.0350
	SG-Elect	0.3763	0.3715	0.373	0.0249
	Electra	0.3138	0.3179	0.3533	0.0225
	GAN-BERT	0.3185	0.327	0.3406	0.0337
	SS-Trainer	0.2176	0.2136	0.2333	0.0111
US Airline	GAN-BElectra	**0.6722**	**0.7239**	**0.7108**	0.0195
	SG-Elect	0.6659	0.7203	0.7072	0.0195
	Electra	0.5493	0.6318	0.623	0.0331

	GAN-BERT	0.6413	0.7054	0.7032	0.0514
	SS-Trainer	0.4178	0.446	0.4307	0.0480
SemEval	GAN-BElectra	**0.5663**	**0.5769**	**0.5741**	0.0145
	SG-Elect	0.558	0.5658	0.5635	0.0228
	Electra	0.3992	0.4309	0.4536	0.0184
	GAN-BERT	0.473	0.478	0.5208	0.0448
	SS-Trainer	0.4072	0.4367	0.4472	0.0305

Table 3: Summary of Results (F1 Macro, F1 Weighted Avg., Accuracy, Standard Deviation)

Dataset	Technique	Mean Standard Error	Confidence Interval (at 95%) of Standard Error (Half Width)
SST5	GAN-BElectra	0.6175	0.0203
	SG-Elect	0.627	0.0202
	Electra	0.6467	0.0199
	GAN-BERT	0.6594	0.0198
	SS-Trainer	0.7667	0.176
US Airline	GAN-BElectra	0.2892	0.0164
	SG-Elect	0.2928	0.0165
	Electra	0.377	0.0176
	GAN-BERT	0.2968	0.0165
	SS-Trainer	0.5693	0.0179
SemEval	GAN-BElectra	0.4259	0.0153
	SG-Elect	0.4365	0.0154
	Electra	0.5464	0.0154
	GAN-BERT	0.4792	0.0155
	SS-Trainer	0.5528	0.0154

Table 4: Summary of Results (Standard Error, Confidence Interval of Standard Error)

Dataset	GAN-BERT Pseudo Label Accuracy	SS-Trainer Pseudo Label Accuracy
SST5	0.3328	0.2255
US Airline	0.7087	0.4229
SemEval	0.5209	0.4403

Table 5: GAN-BERT's accuracy in pseudo label generation across three datasets

Dataset	Technique	Very Negative	Negative	Neutral	Positive	Very Positive
SST5	SS-Trainer	71.36	78.66	76.58	40.53	52.90
	GAN-BERT	28.64	21.34	23.42	59.47	47.10
US Airline	SS-Trainer		4.30	4.62	10.41	
	GAN-BERT		95.70	95.38	89.59	
Sem-Eval	SS-Trainer		9.99	10.39	9.38	
	GAN-BERT		90.01	89.61	90.62	

Table 6: Percentage contribution to final selected pseudo labels for each class by GAN-BERT and SS-Trainer (SG-Elect only)

	SST5	US Airline	SemEval
GAN-BERT	31.50	94.55	90.02
SS-Trainer	68.50	5.45	9.98

Table 7: Percentage contribution to final selected pseudo labels by GAN-BERT and SS-Trainer (SG-Elect only)

3.6.1 SST5 Dataset

Both GAN-BElectra and SG-Elect outperformed GAN-BERT, Electra, and SS-Trainer for the SST5 dataset in mean accuracy. GAN-BElectra also outperformed its predecessor, SG-Elect. Table 3 shows that GAN-BElectra achieved an average accuracy of 0.3825 while SG-Elect, Electra, GAN-BERT, and SS-Trainer scored 0.373, 0.3533, 0.3406, and 0.2333 respectively. Table 3 also contains mean F1 macro and F1 weighted average scores for all techniques and shows that GAN-BElectra and SG-Elect outperform all other evaluated techniques in these two other metrics as well for the SST5 dataset, while SG-Elect was outperformed by its successor GAN-BElectra. Table 8 shows that our proposed solutions also performed better than the other evaluated techniques in F1-score for individual classes. GAN-BElectra outperformed SG-Elect in F1-score for the "Very Negative", "Positive", and "Very Positive" classes, whereas SG-Elect achieved a higher F1-score for the "Negative" and "Neutral" classes. GAN-BElectra also outperformed GAN-BERT and Electra for F1-score for all individual classes except the "Neutral" class. Confusion matrices in Figure 21 visualize the correct and incorrect predictions by all evaluated techniques for the SST5 dataset.

Technique	Class	Precision	Recall	F1-score	Accuracy
GAN-BElectra	Very Negative	0.3246	0.5253	0.397	0.3825
	Negative	0.4493	0.2461	0.3129	
	Neutral	0.2481	0.2601	0.2537	
	Positive	0.4232	0.485	0.4499	
	Very Positive	0.4969	0.4773	0.4847	
SG-Elect	Very Negative	0.3357	0.4854	0.3912	0.373
	Negative	0.4604	0.2755	0.3301	
	Neutral	0.2617	0.3488	0.2944	
	Positive	0.4434	0.3452	0.3849	
	Very Positive	0.4695	0.5072	0.4807	
Electra	Very Negative	0.4121	0.3831	0.2701	0.3533
	Negative	0.3828	0.2062	0.2363	
	Neutral	0.2385	0.3119	0.2663	
	Positive	0.3813	0.4459	0.4063	
	Very Positive	0.5194	0.4514	0.3901	
GAN-BERT	Very Negative	0.3114	0.4908	0.3725	0.3406
	Negative	0.4753	0.2027	0.2476	
	Neutral	0.2358	0.3524	0.27	
	Positive	0.3932	0.3564	0.3539	

	Very Positive	0.4881	0.4159	0.4396	
SS-Trainer	Very Negative	0.1673	0.3432	0.2102	0.2333
	Negative	0.3477	0.2176	0.2357	
	Neutral	0.1932	0.42	0.2633	
	Positive	0.3636	0.0892	0.1429	
	Very Positive	0.3119	0.1959	0.2361	

Table 8: Detailed Results for the SST5 Dataset

3.6.2 US Airline Dataset

Table 9 shows the detailed comparative classification result for GAN-BElectra, SG-Elect, GAN-BERT, Electra, and SS-Trainer for the US Airline dataset. GAN-BElectra and SG-Elect achieved an overall accuracy of 0.7108 and 0.7072 respectively, while Electra, GAN-BERT, and SS-Trainer scored, 0.623, 0.7032, and 0.4307 respectively. This finding is further strengthened by the fact that GAN-BElectra and SG-Elect outperform all other evaluated techniques in the mean F1-macro and F1 weighted average scores as well for the US Airline dataset (see Table 3). GAN-BElectra also outperforms SC-Elect in this metric. As demonstrated in Table V, GAN-BElectra and SG-Elect achieved higher F1-score for all three individual classes compared to GAN-BERT, Electra, and SS-Trainer, while GAN-BElectra also outperforms SG-Elect in the same metric. Figure 22 shows the confusion matrices that visualize the correct and incorrect predictions made by all four evaluated techniques for the US Airline dataset.

Technique	Class	Precision	Recall	F1-score	Accuracy
GAN-BElectra	Negative	0.8896	0.7231	0.7975	0.7108
	Neutral	0.463	0.6516	0.5404	
	Positive	0.6277	0.7408	0.6788	
SG-Elect	Negative	0.8921	0.7202	0.7967	0.7072
	Neutral	0.4686	0.6328	0.5372	
	Positive	0.5932	0.7542	0.6637	
Electra	Negative	0.8282	0.6759	0.7375	0.623
	Neutral	0.4369	0.4882	0.4468	
	Positive	0.398	0.5939	0.4636	
GAN-BERT	Negative	0.835	0.7752	0.7938	0.7032
	Neutral	0.5015	0.5484	0.5048	
	Positive	0.6342	0.6264	0.6253	
SS-Trainer	Negative	0.834	0.3477	0.4876	0.4307
	Neutral	0.2892	0.428	0.333	
	Positive	0.3106	0.7571	0.4328	

Table 9: Detailed Results for the US Airline Dataset

3.6.3 SemEval Dataset

Table 10 demonstrates that GAN-BElectra and SG-Elect outperform GAN-BERT, Electra, and SS-Trainer in terms of classification accuracy for the SemEval dataset. They achieved an accuracy score of 0.5741 and 0.5635 respectively while Electra, GAN-BERT, and SS-Trainer achieved, 0.4536, 0.5208, and 0.4472 respectively. We also observe in Table 3 that GAN-BElectra and SG-Elect outperform all other evaluated techniques in terms of mean F1 macro and F1 weighted average scores as well for the SemvEval dataset. In line with other datasets, GAN-BElectra also outperforms SG-Elect in this metric. As can be seen in Table 10, GAN-BElectra performed better than SG-Elect in 2 out of 3 individual classes, achieving a higher F1-score for the "Neutral" and "Positive" classes while SG-Elect achieved a higher F1-score for the "Negative" class. Both Electra and GAN-BERT underperformed in terms of F1-scores for all individual classes compared to SG-Elect and GAN-BElectra. Confusion matrices in Figure 22 visualize the correct and incorrect predictions made by GAN-BElectra, SG-Elect, GAN-BERT, Electra, and SS-Trainer for the SemEval dataset.

Technique	Class	Precision	Recall	F1-score	Accuracy
GAN-BElectra	Negative	0.4373	0.6046	0.507	0.5741
	Neutral	0.5992	0.5524	0.5727	
	Positive	0.6665	0.5857	0.6191	
SG-Elect	Negative	0.4374	0.6219	0.5116	0.5635
	Neutral	0.5977	0.5244	0.557	
	Positive	0.639	0.5826	0.6055	
Electra	Negative	0.2835	0.3515	0.3006	0.4536
	Neutral	0.5288	0.596	0.5526	
	Positive	0.5434	0.3259	0.3445	
GAN-BERT	Negative	0.4542	0.4787	0.44	0.5208
	Neutral	0.5736	0.5394	0.468	
	Positive	0.634	0.5191	0.5109	
SS-Trainer	Negative	0.2891	0.3573	0.2839	0.4472
	Neutral	0.5116	0.5231	0.4958	
	Positive	0.5199	0.3978	0.4418	

Table 10: Detailed Results for the SemEval Dataset

3.6.4 Ablation Study

Two major components within GAN-BElectra are GAN-BERT and Electra. Additionally, in SG-Elect, another important component is SG-Trainer. As part of our ablation study, we investigated the individual performance of these components on the same test data used for GAN-BElectra and SG-Elect. This analysis provided insight into the effect of individual components that constitute our proposed solution. Table 3 and Table 4 contain the summary of these results for GAN-BERT,

Electra, and SS-Trainer, while Table 8, Table 9, and Table 10 provide detailed results with metrics for each class. Below we discuss the performance of these individual components on the three datasets used with our proposed solutions: SG-Elect and GAN-BElectra.

3.6.4.1 US Airline and SemEval Dataset

GAN-BERT outperformed Electra and SS-Trainer for our two 3-class Sentiment Classification datasets (US Airline and SemEval). For the US Airline dataset, GAN-BERT achieved an overall accuracy of 0.7032 whereas Electra and SS-Trainer achieved 0.623 and 0.4307 respectively. Similarly, for the SemEval dataset, GAN-BERT achieved an overall accuracy score of 0.5208 while Electra and SS-Trainer achieved 0.4536 and 0.4472 respectively.

3.6.4.2 SST5 Dataset

For the SST5 dataset, Electra outperformed GAN-BERT, achieving an overall accuracy of 0.3533 while GAN-BERT scored 0.3406. However, SS-Trainer consistently scored the lowest (0.2333) for this dataset like the other dataset.

3.6.4.3 Pseudo Label Generation

We also investigated the accuracy of pseudo labels generated by GAN-BERT which were eventually utilized to train Electra along with the few original labeled data. For three different datasets we have used (SST5, US Airline, and SemEval), the average accuracy of GAN-BERT's generated pseudo labels were 0.3328, 0.7087, and 0.5209 respectively, as demonstrated in Table 5.

In SG-Elect, SS-Trainer was also used in addition to GAN-BERT for pseudo-label generation. The average accuracy of SS-Trainer's generated pseudo labels were 0.2255, 0.4229, and 0.4403 respectively for SST5, US Airline, and SemEval datasets. We also inspected how each pseudo label generator (GAN-BERT and SS-Trainer) contributed to the final select pseudo labels for SG-Elect. It shows how the contribution differed for 3-class (SST5 dataset) and 5-class (US Airline and SemEval datasets) Sentiment Analysis. For 3-class Sentiment Analysis, GAN-BERT consistently contributed significantly more pseudo labels to the final Electra-based classifier in our architecture, with 94.55% and 90.02% for US Airline and SemEval datasets respectively. In contrast, for 5-class Sentiment Analysis, SS-Trainer contributed 68.50% pseudo labels to the final

classifier. Table 6 shows the percentage contribution to selected pseudo labels by GAN-BERT and SS-Trainer for individual classes across three datasets used in this evaluating SG-Elect.

In the subsequent section, we discuss the results we achieved with our proposed solution along with the insights gained from the ablation study.

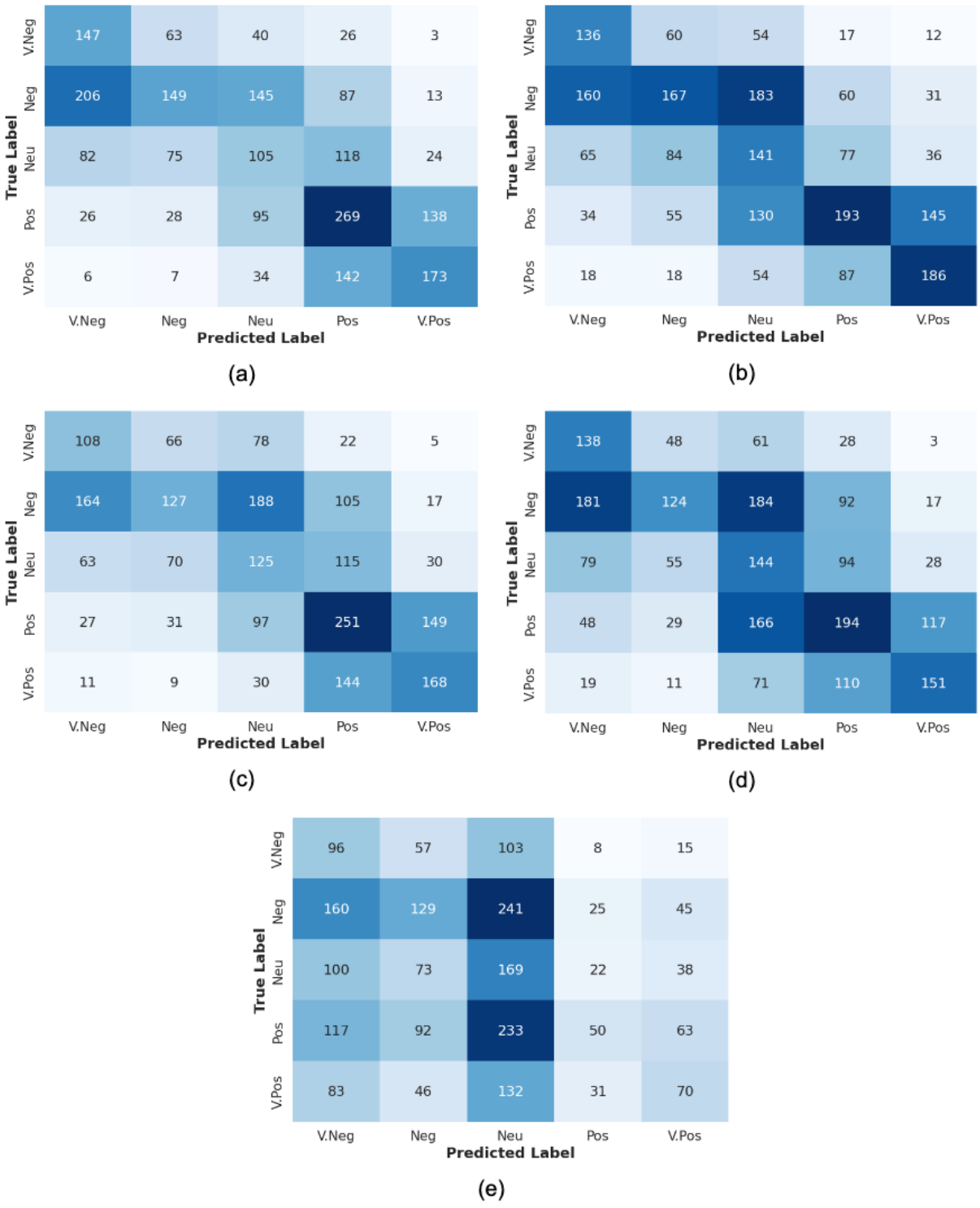

Figure 21: Confusion matrices for SST5 dataset for (a) GAN-BElectra, (b) SG-Elect, (c) Electra, (d) GAN-BERT, (e) SS-Tainer

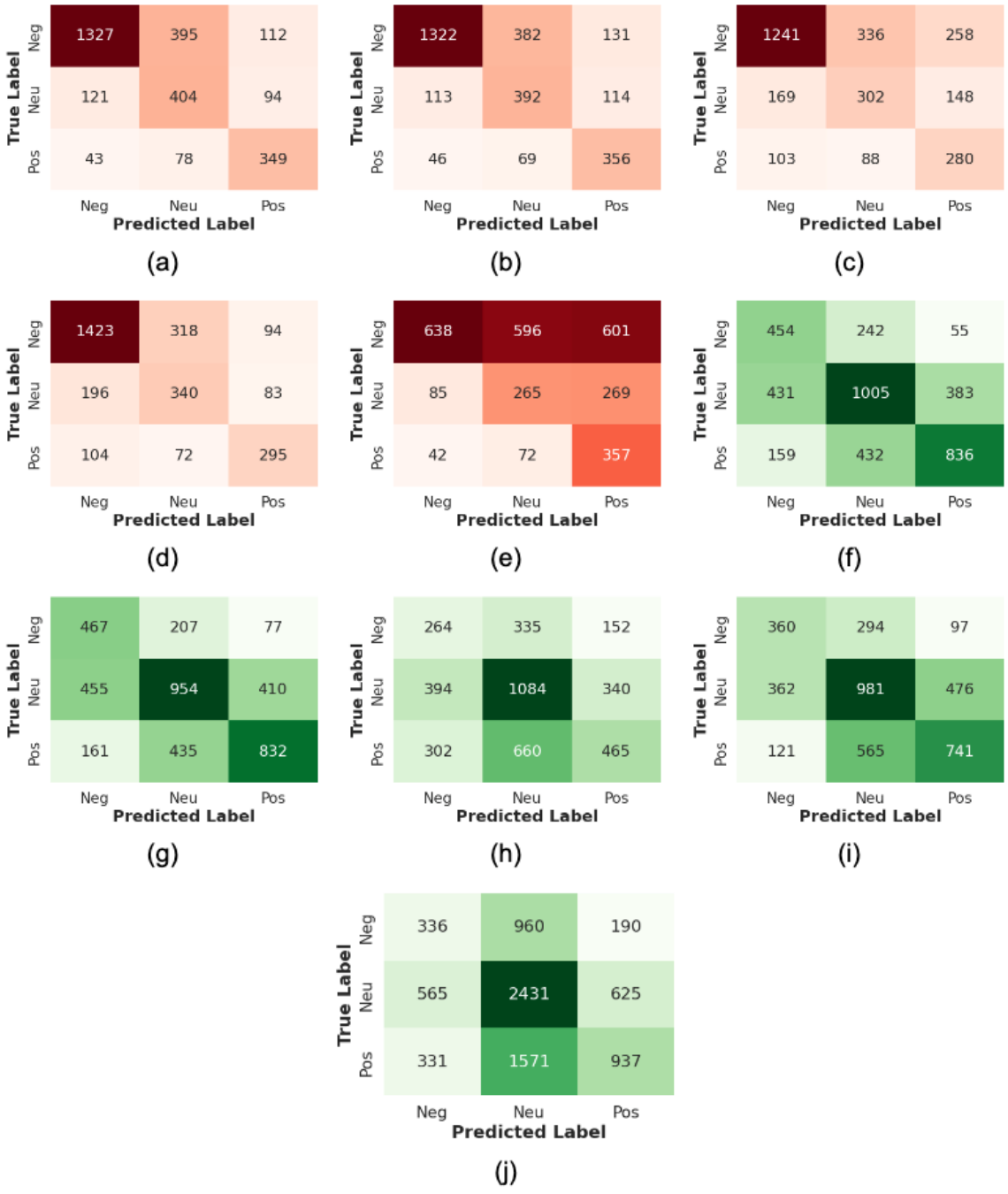

Figure 22: Confusion matrices(with red shades) for the US Airline dataset for (a) GAN-BElectra (b) SG-Elect (c) Electra (d) GAN-BART (e) SS-Trainer and confusion matrices (with green shades) for the SemEval dataset (f) GAN-BElectra (g) SG-Elect (h) Electra and (i) GAN-BART (j) SS-Trainer

3.6.5 Statistical Significance Test

We perform statistical significance test in order to understand if the differences in model performance (in terms of various classification metrics: accuracy, F1 macro, F1 weighted average) are statistically significant. For significance testing, we use the value of these three metrics from each experiment run across all three datasets. Our experimental setup includes three experiment runs (see Section 3.5.1.4) per evaluated model. Each run consists of different train and test split, which we control using seed number. For example, for the SST5 datasets, we had three train and test splits: Train1:Test1, Train2:Test2, Train3:Test3. Each technique was evaluated on these three splits for this dataset, and similarly, for the other two datasets. Consequently, each model was evaluated on a total of nine sets of train and test split. As a result, we have nine related (or paired) data points per metric for the significance testing for model comparison. This means, for example, to test the statistical significance of the difference in F1 macro score achieved by SG-Elect and GAN-BERT, we use nine paired datapoints for each of these models. For each of the three datasets (SST5, US Airline, SemEval), there are three F1 macro scores obtained from three experiment runs for each model, resulting in a total of nine data points for F1 macro per model.

We use Wilcoxon Signed Ranks Test [180] for the significance test, which is often leveraged for comparing two classifiers over multiple different datasets (for relevant guidelines, see [181], [182]; for example usages, see [183]–[185]). Since we had three randomized experiment runs per technique per dataset due to the limited scope of the study, statistical significance test for models' performance comparison for each individual dataset is not feasible due to the small sample size (i.e., 3 datapoints per model per statistical test). Instead, we utilize the Wilcoxon Signed Ranks Test [180] across all datasets together, which avail us with nine data points per model per statistical test. This helps us understand the statistical significance of the differences in the overall performance of the models we evaluated across all datasets, which satisfies our goal of understanding models' overall performance in our selected task.

We use a significance threshold of 0.05 in the Wilcoxon Signed Ranks Test that we conduct. Our null hypothesis (H_0) for these tests is that "*there is no statistically significant difference in the performance of the models in comparison"*. If the p-value is less than 0.05, we reject the null hypothesis, and accept the alternative hypothesis (H_a) which states that *"there is statistically significant difference in the performance of the models in comparison"*.

We compare the performance of SG-Elect and GAN-BElectra with GAN-BERT, Electra, and SS-Trainer. We also compare SG-Elect's performance with GAN-BElectra. Table 11 and Table 12 represent the results from these statistical tests, where we report the asymptotic p-value (2-tailed) from the Wilcoxon Signed Ranks Tests [180] for three classification metrics: accuracy, F1 macro, F1 weighted average.

Based on the accuracy score, SG-Elect's superior performance is statistically significant (p-value < 0.05; i.e., rejects H_0) in comparison with all other evaluated models except GAN-BElectra and GAN-BERT (see Table 11). For the F1 macro score and F1 weighted average, SG-Elect's performance gain is statistically significant (p-value < 0.05; i.e., rejects H_0) compared to all models except GAN-BElectra.

From Table 12, we observe that GAN-BElectra's performance is significantly superior (p-value < 0.05; i.e., rejects H_0) than all the other models except SG-Elect based on F1 macro and F1 weighted average. For the accuracy score, GAN-BElectra's performance is significantly higher (p-value < 0.05; i.e., rejects H_0) than all other evaluated models except SG-Elect and GAN-BERT.

Models in Comparison		p-value (Accuracy)	p-value (F1 Macro)	p-value (F1 Weighted Average)
SG-Elect vs.	GAN-BERT	0.086	**0.015**	**0.038**
	Electra	**0.012**	**0.015**	**0.008**
	SS-Trainer	**0.008**	**0.008**	**0.008**

Table 11: Results (p-values) from significance testing to compare SG-Elect's performance with GAN-BERT, Electra, and SS-Trainer using Wilcoxon Signed Ranks Test [180] for accuracy, F1 macro, and F1 weighted average scores. **Bold** indicates statistical significance (p-value < 0.05).

Models in Comparison		p-value (Accuracy)	p-value (F1 Macro)	p-value (F1 Weighted Average)
GAN-BElectra vs.	SG-Elect	0.214	0.26	0.314
	GAN-BERT	0.051	**0.008**	**0.015**
	Electra	**0.011**	**0.008**	**0.008**
	SS-Trainer	**0.008**	**0.008**	**0.008**

Table 12: Results (p-values) from significance testing to compare GAN-BElectra's performance with SG-Elect, GAN-BERT, Electra, and SS-Trainer using Wilcoxon Signed Ranks Test [180] for accuracy, F1 macro, and F1 weighted average scores. **Bold** indicates statistical significance (p-value < 0.05).

3.7 Discussion

3.7.1 Impact of double fine-tuning

For both SG-Elect and GAN-BElectra, we observe that double fine-tuning Electra, once with pseudo labels and next with a few original labels boosts the accuracy of this transformer network compared to training it only once with either pseudo labels or the original few labels. This confirms the finding in previous studies [186] where researchers noticed a similar improvement in classification accuracy by fine-tuning pretrained models twice.

3.7.2 Performance gain with fewer training steps and reduced network complexity

Our experiments suggest that the semi-supervised self-training sub-component of SG-Elect [162] that included a stacked classifier consisting of a Stochastic Gradient Descent (SGD) Classifier [187] and an XGBClassifier [159] negatively contributed to the overall classification accuracy achieved by SG-Elect. SG-Elect's successor, GAN-BElectra, removes that component from the architecture along with SG-Elect's "Combinator" component. We observe that this did not adversely impact the performance. Moreover, as demonstrated in in the reported results, GAN-BElectra achieved a small gain in performance (arithmetically) over SG-Elect for all the evaluated datasets for all three evaluation metrics (see Table 3). While GAN-BElectra's performance gain over SG-Elect is small and statistically insignificant (see Table 12), the gain is consistent across all evaluated datasets, and we consider this as an additional improvement over its predecessor while acknowledging its lack of statistical significance (some examples of similar considerations in the literature where significance test was not carried out or reported: [155], [175], [188], [189]). It is also important to emphasize that compared to SG-Elect, GAN-BElectra reduces the complexity of the architecture as it relieves the need for training an additional machine learning component that consumes resources and incurs more training cycles.

3.7.3 Impact of pseudo labels on the final result

For SG-Elect, where we used SS-Trainer as one of the pseudo label generators along with GAN-BERT, we noticed that despite having low accuracy, SS-Trainer still had a high confidence score for its prediction for the SST5 dataset, resulting in more of its generated pseudo labels to be filtered

in by SG-Elect's Combinator component which selected the best pseudo labels based on confidence score. We argue that if SS-Trainer was well calibrated (i.e., SS-Trainer's confidence score was in sync with its classification accuracy for SST5), GAN-BERT would have had a larger share in the final selected pseudo labels by the Combinator. This indicates that SS-Trainer had a positive bias in its probability score. It is particularly more apparent for the SST5 dataset. A worthwhile future investigation is to introduce the model calibration process [190] for SG-Elect's pseudo label generators, which can potentially mitigate such unanticipated positive bias.

We also observed that for US Airline and SemEval datasets, the overall accuracy scores of SG-Elect were 0.7072 and 0.58875 respectively for these two datasets and SS-Trainer contributed only 5.45% and 9.98% to final pseudo labels for them respectively. From these observations, we deduce that the higher the contribution of SS-Trainer in the selected pseudo labels, the lower the accuracy our architecture achieved. We evaluated this hypothesis with GAN-BElectra, where GAN-BERT is the sole pseudo label generator, resulting in higher overall accuracy for all three datasets compared to SG-Elect. As reported in the results, we notice that the accuracy of the generated pseudo labels by GAN-BERT (0.3328, 0.7087, 0.5209 for SST5, US Airline, and SemEval respectively) resembles the final results of GAN-BElectra for all three datasets (0.3825, 0.7108, 0.5741 in the same order). This highlights the impact of the GAN-BERT as a pseudo-label generator in GAN-BElectra's overall architecture, making it an important component of the proposed solution.

Without the contribution of GAN-BERT as a pseudo-label generator, the performance of GAN-BElectra would have degraded significantly, as can be inferred from our ablation study where we tested the performance of the stand-alone Electra pre-trained model fine-tuned only on the original few labels. In terms of average accuracy, this fine-tuned stand-alone Electra model achieves 0.3533, 0.6233, and 0.4536 for SST5, US Airline, and SemEval dataset respectively, whereas our proposed model GAN-BElectra demonstrates a boost in performance contributed by the pseudo label generator GAN-BERT, and achieved 0.3825, 0.7108, 0.5741 in terms of average accuracy for these three datasets in the same order. Although slightly lower than GAN-BElectra, a similar boost in performance was observed for SG-Elect as well, which utilized both GAN-BERT and SS-Trainer as pseudo-label generators. It achieved average accuracy scores of 0.373, 0.7072, and 0.5635 for SST5, US Airline, and SemEval datasets respectively.

For SG-Elect, we also notice that, with SST5, GAN-BERT had a significantly higher share of selected pseudo labels for the Positive and Very Positive classes compared to the other three classes. The opposite is observed for the US Airline dataset where SS-Trainer had a larger share of pseudo labels for the Positive class compared to the other two classes. These differences do not seem to have a noticeable impact on the overall accuracy achieved by SG-Elect.

3.7.4 Prediction trends for individual class

3.7.4.1 Incorrect predictions more likely to fall into the adjacent categories

We observe that the majority of incorrect predictions typically fall into the adjacent sentiment categories. This means that a "positive" datapoint is more likely to be predicted as "positive" and "neutral" (also "very positive" for the 5-class dataset) compared to contrasting labels such as "negative". For instance, for the SST5 dataset, out of a total of 556 datapoints with the "positive" label, GAN-BElectra correctly predicted the label for 269 datapoints as "positive", and then incorrectly predicted 138 datapoints as "very positive", 95 datapoints as "neutral", only 26 datapoints as "very negative" and 28 as negative. This similar phenomenon has been observed for almost techniques including SG-Elect, with all three datasets (see confusion matrices in Figure 21 and Figure 22).

3.7.4.2 Variation in pseudo label accuracy

The accuracy of pseudo label generation by GAN-BERT and SS-Trainer (only applicable for SG-Elect) varied significantly across our three selected datasets. GAN-BERT achieved 0.3328, 0.7087, 0.5209 in pseudo label generation accuracy for SST5, US Airline, and SemEval datasets respectively, and SS-Trainer achieved 0.2255, 0.4229, and 0.4403 (in the same order). Since SST5 is a finer-grained sentiment dataset (i.e., 5 sentiment classes whereas the other two datasets have 3 classes), we expected it to contribute to lower classification accuracy compared to US Airline and SemEval dataset. This is indeed consistent in all other experiments performed in this study.

We also observed that between US Airline and SemEval datasets, though they are both 3-class sentiment datasets, GAN-BERT scored 0.7087 for the accuracy for the former compared to 0.5209 for the latter. We believe this is due to the different compositions of the two datasets. For example, the US Airline dataset has a significantly higher number of "negative" datapoints compared to the other two classes. On the other hand, SemEval contains much fewer "negative" datapoints

compared to the other two classes. However, Figure 19 and Figure 20 show that this class imbalance in the SemEval dataset is visibly less significant compared to the US Airline dataset. Confusion matrices in Figure 22 for test data show that all evaluated techniques performed more accurate predictions for the neutral class for the SemEval dataset compared to the other two classes.

For the US Airline dataset, it is the negative sentiment class where more accurate predictions occurred across all techniques. Based on these observations, we argue that the different compositional attributes of these two datasets (e.g., a significantly higher number of "negative" datapoints in the US Airline dataset compared to the SemEval dataset which was visibly less imbalanced) might had an impact on the overall differences in the accuracy of pseudo label generation for these two datasets for GAN-BERT. As mentioned in earlier, our rationale for not artificially balancing the composition of the datasets was due to the fact that the primary training data used in our experiments were organically balanced since we used 50 datapoints for each class for each dataset in each of our experiments as the "few original labeled training data". In addition, along with accuracy, we also reported the F1-macro score which provided a more reliable comparison of techniques when class imbalance is present in the selected dataset. However, we believe that this may still had an impact on the pseudo label generation accuracy as well as the variations in the number of correctly predicted labels for our test data (see Confusion matrices in Figure 21 and Figure 22). In contrast, in SG-Elect, where SS-Trainer was also used as a pseudo label generator along with GAN-BERT, this phenomenon was not observed. Instead, SS-Trainer had similar performance in pseudo label accuracy for both 3-class datasets, scoring 0.4229 and 0.4403 for US Airline and SemEval respectively. This provides an opportunity for future investigation to inspect the underlying reason for this difference in performance, particularly considering the fact that SS-Trainer is a traditional machine learning technique whereas GAN-BERT is based on deep learning technologies.

It is also noteworthy that between GAN-BERT and Electra, the two main components of GAN-BElectra, GAN-BERT outperforms Electra for two 3-class sentiment datasets, while Electra outperforms GAN-BERT for the SST5 dataset which contains 5-class sentiment data. Fine-tuning stand-alone Electra is significantly faster than fine-tuning GAN-BERT since GAN-BERT uses a semi-supervised training method, which essentially performs many iterations of supervised training with varying training data. Despite this, Electra achieving higher accuracy for SST5 than GAN-BERT is an interesting observation that warrants further investigation.

3.8 Conclusion

In this chapter, we discuss SG-Elect and GAN-BElectra, two successive techniques for multi-class Sentiment Analysis with limited labeled data. Our experiments suggest that SG-Elect achieves significantly higher performance (i.e., F1 macro, and F1 weighted average) in multi-class Sentiment Analysis compared to its baseline (GAN-BERT). We subsequently develop GAN-BElectra, that builds upon SG-Elect and significantly reduces the architecture complexity and required training steps compared to SG-Elect, without having any adverse effect on the performance. It is noteworthy that GAN-BElectra achieved a small gain in performance (arithmetically) consistently across all datasets we evaluated though the gain in performance did not reach statistical significance (see section 3.6.5 and 3.7.2 for more details).

Though GAN-BElectra demonstrated better performance compared to its predecessor (i.e., SG-Elect), the margin of improvement is not statistically significant. This suggests that there is still room for further improvement in this area.

We performed statistical significance testing across all datasets to compare the performance of our proposed architectures over other evaluated techniques. It can be a worthwhile investigation to conduct similar statistical significance testing for each individual dataset which would require conducting more experiment runs (i.e., 10 or more) per models for each dataset than we performed in our study (i.e., three).

Another noteworthy limitation of both the proposed solutions is that they require a multi-step training process involving training the pseudo label generator(s) first and then training the Electra component. Achieving an end-to-end solution without requiring a multi-step training process is a worthwhile future exploration.

Chapter 4: Emotion Cause Generation: A New NLP Task

From Sentiment Analysis discussed in the previous chapter, this chapter extends our focus on human emotion towards the more complicated tasks related to the cause of emotion. This is the focal point of a relatively new NLP research area, named Emotion-Cause Analysis (ECA). ECA has recently garnered substantial attention from the researcher community. In addition to devising various techniques to solve ECA-related tasks, researchers also introduced different variants of the ECA tasks such as Emotion Cause Extraction (ECE), Emotion Cause Pair Extraction (ECPE), Emotion Cause-Pair Span Extraction (ECSP). These are primarily classification tasks where the cause of the emotion and/or type of emotion expressed in the text are identified. In this study, we propose the first-ever generative NLP task within the ECA domain, named Emotion Cause Generation (ECG). The objective of this task is to generate a meaningful cause for an emotion expressed in a given text. We demonstrate the viability of this newly proposed task with promising early observation.

The work presented in this chapter has been published in one conference[3].

4.1 Introduction

Recent advances in high-performance computing coupled with omnipresent cloud technologies have been a catalyst in amplifying the volume of research work performed many in domains. Machine learning and its various applications are prime examples of such domains. A plethora of recent research now focuses on leveraging machine learning to perform various tasks, such as Sentiment Analysis (e.g., [191]–[193]), Emotion-Cause Analysis [90], [194], [195], persuasion (e.g., [196]–[198]), predicting future events, (e.g., [199]) image generation from the text (e.g., [200]). Machine learning is also being utilized in many industries on a routine basis and it is contributing to novel consumer products such as autonomous cars, personalized virtual assistants, etc. [135]. Natural Language Processing (NLP) is an interdisciplinary research area that has seen extraordinary growth in recent years influenced by the advancement in machine learning technologies along with the availability of a large amount of data. NLP is typically broken down

[3] Riyadh, M., & Shafiq, M. O. (2022). Towards Emotion Cause Generation in Natural Language Processing using Deep Learning. In 21st IEEE International Conference on Machine Learning and Applications (ICMLA), 2022.

into two branches: NLU (Natural Language Understanding) and NLG (Natural Language Generation) [1]. These two subdomains of NLP complement each other which eventually constitute the overall domain of NLP.

Within NLP, there are a plethora of "tasks" that researchers focus on such as Sentiment Analysis, Emotion Cause Extraction (ECA) related tasks. These are primarily some examples of NLU tasks, which have historically been the dominant subarea within NLP. Recent advances in "language models" such as GPT-2 [128] have garnered remarkable focus from the researcher community on the NLG tasks. In this study, we attempt to leverage the advances in general-purpose language models in order to perform the new NLG task we introduce within the domain of ECA, named Emotion Cause Generation (ECG). Figure 23 shows how ECG fits within the existing ECA tasks.

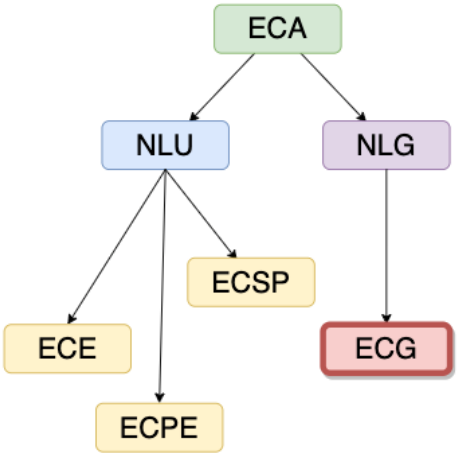

Figure 23: ECG and its relationship with the existing ECA tasks.

There are several tasks within the domain of ECA. The most common among them is ECE (Emotion Cause Extraction). Introduced in 2010 [77], ECE aims to identify the cause of a certain emotion expressed in a given text. Subsequently, many other researchers proposed various rule-based techniques and traditional machine learning techniques to perform this task (e.g., [9], [74], [201]). More recently, some studies proposed more variants of the ECE tasks leveraging deep learning technologies. For instance, Xia et al. introduced the ECPE (Emotion Cause Pair Extraction) task which not only extracts the emotion-cause but also identifies the emotion expressed [90]. Following this, Wei et al. proposed the first end-to-end model for the ECPE task [91]. Li et al. proposed the ECSP (Emotion Cause-Pair Span Extraction) task which aims to

identify the cause of the emotion in the given text at a more granular span-level instead of typical clause-level identification [194].

Though we have witnessed the rapid growth of generative tasks within the broader domain of NLP, such as open-ended text generation, poetry generation, machine translation, and question-answering, the application of generative machine learning has largely remained unexplored in the ECA research area. We propose leveraging text generation methods to generate meaningful causes of an emotion expressed in a given text where the "cause of the expressed emotion" is missing. We name this task – Emotion Cause Generation (ECG). We believe that this new task and its potential extensions can be useful in various applications. For instance, an automated mechanism to generate meaningful causes for an emotion expressed in a given context can be useful in specialized artificial intelligence systems where it needs to offer suggestions as to why someone may be experiencing certain emotional states. In addition, this task initiates the exploration of generative machine learning in the ECA research area. This initiative is expected to encourage further similar exploration in this area, planting the seed for many impactful future research activities.

In this study, we particularly demonstrate the ECG task a text-infilling NLP task. We build upon an existing infilling machine learning solution [202] to perform the ECG task to demonstrate its viability as a new NLP task in the domain of ECA.

4.2 Contributions

The specific contributions of the study discussed in this chapter include the following:

- We propose a new ECA-related NLP task named Emotion Cause Generation (ECG). This is the first ever generative NLG task (i.e., NLG task) in the ECA domain.
- We establish the viability of the proposed ECG task as an NLP task through technical demonstration that includes:
    - Identifying and adapting an existing ECA-related dataset [9] for the proposed ECG task.
    - Building upon an existing text generation infilling solution [202] to perform the ECG task.

- Leveraging various existing quantitative methods to evaluate the generated causes coupled with qualitative analysis performed by the authors.

4.3 Related Studies

Lee et al. [77] first proposed the ECE task. The objective was to find the underlying cause of a specific emotion that is conveyed in a given text. In contingency to their work, many researchers have subsequently designed various techniques based on traditional machine learning and rule-based based methods to tackle this problem [9], [78], [203]. Despite their good performance, these methods were still highly dependent on the complex feature design and suffered from poor learning ability compared to their successors such as deep learning methods.

The recent advance in deep learning has inspired some researchers to put their focus on discerning the cause of emotions by leveraging neural networks along with the attention mechanism in order to discern the underlying cause of an emotion. Gui et al. [74] issued a new dataset in the Chinese language with texts containing various expressed emotion and their causes and devised a technique to identify the causes of emotions. Leveraging this corpus, Gui et al. [87] employed a machine learning model that utilizes a deep memory network to identify the cause of emotion by leveraging a question-answering model. Chen et al. [204] developed a technique for detecting emotion-cause in microblogs with a hierarchical convolution neural network. It uses an encoder at the clause-level and another at the subtweet-level to integrate contextual features. Xu et al. [205] adopted a lesson to re-categorize methods based on a set of features dependent on emotion along with a set of emotion-independent features.

Xia et al. [206] offered a new ECPE task within ECA where the goal was to also determine the emotion expressed in the sentences in addition to identifying the associated cause-clause of the emotion. To evaluate their proposed task, they enhanced the datasets released by Gui et al. [74]. Fan et al. [207] reconstructed the ECPE task into a technique of directed graph construction by leveraging this newly reconstructed dataset along with BERT [120]. These studies however considered ECA as a clause-level classification task, largely disregarding the reality that cause-clause itself is not the actual cause. Instead, the clause is a set of text that contains the emotion-cause along with other texts. To address this, Li et al. [194] proposed a new ECSP task where they released a novel span-based joint learning approach incorporating BERT [120] and attention

mechanism [95] as the backbone of their technical solution. However, these new tasks still confined ECA under NLU, without any indication of how NLG can be useful for ECA.

With the introduction of the Transformer [95], particularly BERT [120], the NLP research area, specifically NLG, has seen exponential growth, primarily towards large pre-trained language models (LLMs). These pretrained LLMs have demonstrated an outstanding ability to generate text with few-shot and zero-shot learning [129], [208]. LMs typically consist of a large number of parameters with some having several hundred billion parameters [208]. The highest increase in the model size (117M to 500B) came within Auto-regressive LLMs [128], [208]. GPT-2 [128] first brought vast enhancement to the quality and articulacy of the general-purpose LLMs. This trend continued with GPT-3 [129] and newer LLMs. Since the introduction of these large LLMs, researchers have proposed many new NLP tasks [128], [129], [209].

One such task is text infilling which attempts to "fill in the blank" by predicting missing pieces of a given text. This type of infilling task requires the predicted text to be semantically appropriate for both the preceding and succeeding text. The existing off-the-shelf open-source LLMs such as GPT-2 can effectively produce coherent text [210], [211], but cannot properly perform the infilling operation as they only leverage the text in a single direction as context, typically the preceding ones. In contrast, bidirectional attention-based mechanisms such as BERT [120] and SpanBERT [212] are capable of performing the infilling task by taking both the preceding and succeeding texts into consideration. Nevertheless, this very bidirectional attention capability bounds their ability to infill only to spans with fixed lengths.

Donahue et al. [202] provided a general and flexible infilling framework that can be utilized for various domains. Simply put, their technique takes "text with blanks" as input and generates completed text as output. To achieve this, they first constructed training samples by masking random spans in the input text to generate two input components: text with blanks and the associated answer for each blank. They trained unidirectional LLMs such as GPT-2 on these two input components, joined together. In inference time, the trained model can predict the "answer for the blanks" when the "text with blanks" is given as input. Due to the various relevant features of this framework, such as the ability to infill for variable length spans, the capability to leverage pretrained LLMs and their standard fine-tuning processes, we build upon this framework to

demonstrate the feasibility of the proposed ECG task as an NLP task, particularly, an infilling task under the NLG branch.

4.4 ECG: The Proposed Task

In this study, we propose Emotion-Cause Generation (ECG), the first ever generative NLP task within the ECA domain that aims to generate a meaningful cause for an emotion expressed in a given text. Compared to the existing ECA tasks which primarily focus on extracting the cause of the emotion from a set of given texts or clauses, ECG focuses on generating the text span or n-gram that represents a meaningful cause for the given emotion (Figure 24).

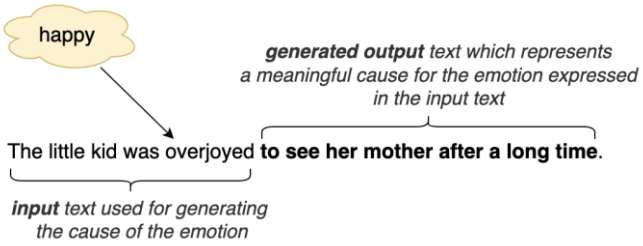

Figure 24: A sentence expressing an emotion ("happy") with the cause indicated in bold.

We regard ECG as an infilling assignment [202]. This means for a given emotion-cause containing text, the cause of the emotion is the “blank” that needs to be “filled” or “generated”. Figure 25 shows an example of such a text with an expressed emotion and a meaningful cause. ECG task aims to generate the text span (indicated in bold in Figure 24) that forms the cause of the emotion for a given input text.

4.5 Demonstration of the Proposed Task

4.5.1 Cause-generation model

To generate a meaningful cause for given emotion-containing text, we treat the emotion-cause generation task as an infilling task. We leverage Ilm [202], which offers a convenient mechanism to fine-tune GPT2 [128] for such infilling tasks. One can mask any part of a sentence, regardless of the role of that part within the sentence (i.e., it can be a random part of a sentence, the verb clause or subject clause etc.), and use Ilm to “infill” that part. In order to use Ilm to generate the specifically masked “cause” clause for the proposed ECG task, we make some modifications to its algorithm and fine-tune it with our selected ECA dataset [9]. We describe these below.

4.5.2 Customization of Ilm for the ECG task

Ilm includes a mechanism called “masking”, with which, the full or a portion of the given text input is omitted and eventually replaced with the generated texts. Ilm offers several masking mechanisms such as the masking of words, entire document, or randomly selected n-gram spans. The masked texts are treated as the “blanks” in the infilling task where Ilm generates appropriate text in place of the masked text.

In order to perform our proposed ECG task, we have created a custom masking step leveraging Ilm’s existing random n-gram masking capability. In Ilm’s existing random n-gram masking functionality, it randomly selects a starting and ending point for the texts to be masked. We build upon this random n-gram masking mechanism to define a new custom masking mechanism where the n-gram’s starting and ending point can be defined based on the identifier within the input text. The specified n-grams represent the cause-spans in our study. We discuss this further in the dataset subsection below.

4.5.3 Fine-tuning

The author of Ilm fine-tuned GPT2 on three datasets and made those trained models publicly available (i.e., one model trained on story excerpts, one on song lyrics, and another on research abstracts). Instead of fine-tuning GPT2 from its original checkpoint, we start fine-tuning from Ilm’s checkpoints as it comes with the understanding of the masking provided within Ilm for the infilling task. During the development phase, we used some random samples from our selected dataset [9] to observe how these different Ilm models may potentially perform the ECG task. We noticed that the Ilm model that was trained on a dataset of story excerpts functioned closer to our use-case compared to the other two datasets: research paper abstracts and song lyrics [202]. We believe this is due to the fact that story excerpts can contain more content related to human emotion in a regular prosaic format compared to the other two types of datasets used to train the other two variations of the Ilm model. We utilize most of Ilm’s original hyperparameters as our analysis during the development phase demonstrated that these default settings are sufficient for our intended task. However, we adjusted a few of these hyperparameters to suit the goal and the scope of the study, which includes capping the max-training step at 1000 and epochs at 500 to optimize the training. After loading the selected Ilm checkpoint, we further fine-tune it with our dataset [9] leveraging our custom masking step.

4.5.4 Inference

After the fine-tuning process as described in the prior sections, the model can be used for inference – to generate a "cause" for a given emotion expressed in a text. The expected input text expresses an emotion with the cause of the emotion "masked" with "an underscore" (i.e., "_"). The fine-tuned model generates the "cause" of the emotion expressed in the text by replacing the mask (i.e., the "underscore"). Figure 25 demonstrates an example of the expected input and output of the fine-tuned model.

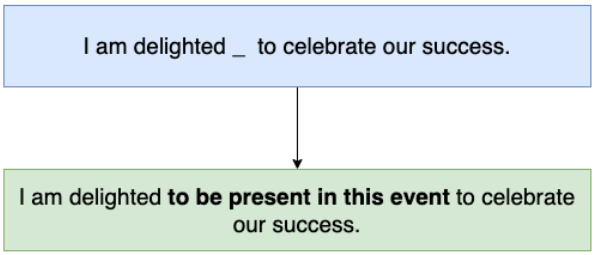

Figure 25: The top sentence represents an input to the model which "masks" the cause of the emotion with an "underscore". The bottom text represents the output text with a generated cause which replaced the masked portion of the input sentence.

4.5.5 Hardware

We leveraged cloud computing resources with Google Colab Pro, which provides elastic computing resources based on processing and storage needs. Typically, the following specifications were the maximum available resources available for us to use at Google Colab:

- CPU: Intel Xeon 2.30GHz with max. 40 cores
- Memory: 36 GB
- Storage: 226 GB

4.6 Dataset

We use the emotion-cause dataset offered by Ghazi et al. [9]. It properly fits our use-case for the demonstration of the ECG task as the causes within the data points are separated as spans using specific tags in the XML format.

4.6.1 Sample Datapoint

Below is an example of a given datapoint of this dataset:

<happy> These days he is quite happy ***<cause>*** *traveling by trolley* ***<\cause>.*** *<\happy>*

The tag "<cause>" indicates the start of the cause-span while "<\cause>" indicates the end of it. The emotion expressed in the datapoint wraps the text with tags of a similar format. In this case, the emotion expressed is "happiness" and the sentence is wrapped with starting (i.e., <happy>) and closing (i.e., </happy>) tags indicating the emotion.

We transform each datapoint to adhere to the expected format in the model used in this study. For example, the above sample datapoint was transformed into the following format:

happy

These days he is quite happy ***<cause>*** *traveling by trolley* ***<\cause>***

In this transformed format, each datapoint comprises of two lines. The first line represents the emotion expressed in the given sentence, and the second line contains the original datapoint with the "cause" tag while removing the "emotion" tags that wrapped the sentence in the original dataset. Our custom masking function leverages the "cause" tags to programmatically identify the "cause span" (i.e., masked n-gram) of the input sentences.

4.6.2 Overall Composition:

The dataset contains 2414 sentences. Among them, there are 820 sentences with both a cause and an emotion tag. In this study, we leverage the datapoints that include both cause and emotion tags. Each sentence in this subset belongs to one of the following seven emotions:

- Happiness
- Fear
- Anger
- Disgust
- Surprise
- Shame
- Sadness

4.6.3 Data Splits:

The datapoints are split into training and validation sets in an 80:20 ratio with 10 reserved datapoints for inference to test the generation capability of the fine-tuned model. The training

dataset was used for fine-tuning the pretrained model. We then used this fine-tuned model to generate the causes for 10 test datapoints, which we discuss below.

4.7 Results

We use our fine-tuned model (i.e., Ilm story excerpt model with custom masking function) to generate cause for each of the 10 test datapoints in our study. As discussed earlier, we mask the "cause" span from these datapoints with underscore ("_") and tasked the fine-tuned model to replace the masks with relevant causes. Table 13 demonstrates the generated text spans (in bold) that represent causes for the emotion expressed in the example inputs. We also include the associated texts with the original causes along with the emotion expressed for ease of inspection for the readers.

#	Emotion	Text with Original Cause	Text with Generated Cause
1	Anger	"I don't feel resentful **that God has decided to take me before my allotted span**."	I don't feel resentful **of what people are up to.**
2	Fear	"In 1986 the local authority became concerned **about injuries which D had sustained** and they obtained a place of safety order."	In 1986 the local authority became concerned **at suspected dog attacks** and they obtained a place of safety order.
3	Sad	"It's possible to be sad **about your babies becoming children**, and yet accept the fact that you won't have any more."	It's possible to be sad **at the dead body of Tamar**, and yet accept the fact that you won't have any more.
4	Fear	"To see him fumbling with our rich and delicate language is to experience all the horror **of seeing a Svres vase in the hands of a chimpanzee."**	To see him fumbling with our rich and delicate language is to experience all the horror **of being swallowed, and to be swallowed by two people, and to feel the sting of it all**.
5	Anger	"They were cross **with each other** because of him, and now Nanny was saying bad things about Smallfry that would make Buddie very, very angry."	They were cross **at him** because of him, and now Nanny was saying bad things about Smallfry that would make Buddie very, very angry.
6	Surprise	"His last illness was the most violent, and his doctors were astounded **that he survived it."**	His last illness was the most violent, and his doctors were astounded **to see him.**
7	Sad	"I was quite sad **to hear my right hon. Friend the Minister of Agriculture, Fisheries and Food say that he was prepared to sit in the negotiations for months."**	I was quite sad **at being unable to come to terms with my father.**

8	Fear	"France expressed horror **at the assassination** and appealed for unity to assure peace in the country it once ruled."	France expressed horror **at the barbarism of yesterday's attack** and appealed for unity to assure peace in the country it once ruled.
9	Happy	"I'm really happy **in the group now."**	I'm really happy **that my mum's gone now.**
10	Happy	"Looking anything but delighted **at the prospect**, he stood up."	Looking anything but delighted **at having his cake and eat it**, he stood up.

Table 13: Texts with original causes and texts with generated causes based on the test data from Ghazi et el. [9] Original texts are within quotation marks, and cause spans are indicated in bold for both original and generated ones.

We inspect these generated causes in the following section.

4.8 Observation and Analysis

4.8.1 Evaluation Procedure

In order to evaluate the generate causes for their meaningfulness in related to the emotion expressed in the sentence, we follow two main approaches:

Qualitative:

We manually inspect each of the generated causes in order to understand its meaningfulness within the context of a given sentence. We report this analysis in section 4.8.2.

Quantitative:

We use different types of quantitative metrics to assess the meaningfulness of the generated causes in relation the emotion expressed in the given text. We report this analysis in section 4.8.3. We leverage some common quantitative metrics that are used for evaluating machine generated texts with reference to the original text. We use them as complementary metrics as they are more suitable for more structured generative tasks (i.e., such as machine translation which can have an approximate verifiable answer) as opposed to the open-ended tasks such as the proposed ECG task where a single or an approximate "correct" answer does not exist. We briefly introduce these metrics below:

- BLEU [213] is a common machine translation evaluation metric that compares the n-gram matches in machine-generated text to those in reference text. BLEU-1 [213] and BLEU-2 [213] measure similarity at the unigram and bigram level respectively.

- METEOR [214] is another metric that combines precision, recall, and synonymy to evaluate translation quality.
- ROUGE [215] is a set of metrics used for text summarization, with ROUGE-1 [215] and ROUGE-2 [215] measuring similarity at the unigram and bigram level respectively, and ROUGE-L [215] using the longest common subsequence.
- BERTScore [216] evaluates text quality by comparing contextualized word embeddings.

As an additional quantitative analysis, we analyze the general linguistic quality of the generated causes and compared them with their original counterparts in the dataset. The following metrics were used for assessing the linguistic quality:

- For readability, we use Flesch Reading Ease metric [217] and Coleman Liau Index [218], both of which deem a text easily readable when it is easy to read for 13-15 years old who are proficient in English. For Flesch Reading Ease metric, this threshold is a score of 60 or above. For Coleman Liau Index, this threshold is a score 8 or below. We report the average score from 10 test datapoints.
- We also report the overall count of grammatical errors within the original and generated sentences.
- Additionally, we report GRUEN score [219] for evaluating grammaticality, non-redundancy, focus, structure and coherence of generated text. It is a value between 0 to 1 (higher is better).

Lastly, we perform sentiment (using "twitter-roberta-base-sentiment" [220]) and emotion classification (using "emotion-english-distilroberta-base" model [221]) of the original and generated causes in order to check their agreement with each other and with the original emotion labels from the dataset.

We discuss our analysis on the generated causes for the 10 test datapoints using these various methods in the following subsections.

4.8.2 Manual Inspection of the Generated Causes

Based on our observation, the generated cause of item# 1 in Table 13 seems to be grammatically and semantically correct. We believe it represents an appropriate cause for the emotion expressed in the given text. We share the same remarks for item# 2 as well. For item# 3 and 4, while the

generated causes sound somewhat bizarre, it still seems to represent the emotions being expressed – "sad" and "fear". They also demonstrate grammatical and semantical correctness.

In our opinion, the generated cause for item #5 seems to suffer from redundancy and lack meaningfulness as a representative cause for the expressed emotion. The generated cause for item# 6 seems to be an appropriate cause for the expressed emotion, while also being grammatically and semantically accurate. For item# 7, we also believe that the generated cause represents the emotion expressed in the given text. It also shares a similar grammatical and semantical quality to most other items. Similar remarks can be made about item# 8 as well.

For item# 9, although the generated cause sounds peculiar, similar to items 2 and 3, it still seems to be a meaningful cause for the expressed emotion while also being grammatically and semantically correct. We believe that the generated cause for item# 10 is also a meaningful cause for the expressed emotion in the text and that it is grammatically and semantically correct.

Overall, with a few exceptions, we opine that the generated causes seem to be meaningful for the associated emotions in the given texts. The consistent grammatical and semantical correctness of the generated texts is also something noteworthy.

4.8.3 Quantitative Analysis

4.8.3.1 Methods that use original text as reference:

We have leveraged several computational methods to understand the general quality of sentences with generated causes in contrast with sentences with original causes. These methods evaluate the generated text by using the original text as a reference. While some of these methods were originally meant for evaluating the quality of machine translation, they have been used to evaluate other natural language generation tasks due to their generalizability [222].

Our test sentences with generated causes achieved a BLEU-1 score [213] of 0.6647. The mean precision for BLEU-1 was 0.6825. BLEU-1 brevity penalty was 0.9739 and the length ratio was 0.9742. Resembling BLEU-1 scores, for BLEU-2, the score was 0.6191, precisions were 0.6825 and 0.5921 (respectively for 1-gram and 2-gram) while brevity penalty and length ratio are expectedly the same with BLEU-1.

We also calculated the METEOR score [214], where our test sentences with generated causes achieved a score of 0.6550. We calculated 3 different variations of ROUGE scores [215]: ROUGE-

1, ROUGE-2, and ROUGE-L, where our test sentences with the generated causes achieved 0.6617, 0.5650, and 0.6580 respectively. Finally, we also evaluated our test sentences with the generated causes for BERTSCORE [216], achieving a score of 0.7905. These results are reported in Table 14.

Metric	Score
BLEU-1 [213]	0.6647
BLEU-2 [213]	0.6192
METEOR [214]	0.6550
ROUGE-1 [215]	0.6617
ROUGE-2 [215]	0.5650
ROUGE-L [215]	0.6579
BERTSCORE [216]	0.7905

Table 14: Evaluation of the generated causes with methods that use original text as a reference

4.8.3.2 Methods to evaluate linguistic quality independent of the reference or original text:

We have also performed quantitative analysis of the generated causes using methods that evaluate the linguistic quality of texts independently considering various factors such as grammatical correctness, readability, etc. We performed two comparative readability tests between the sentences with the original causes and sentences with the generated causes namely: Flesch Reading Ease [217] and Coleman Liau Index [218].

For Flesch Reading Ease, the sentences with the original causes score 63.19 whereas the sentences with the generated causes scored 70.84, indicating relatively similar readability quality of both. As seen in Table 15, for Coleman Liau Index, sentences with generated causes achieved 8.06, while the sentences with the original causes scored 9.63. This also indicates how comparable sentences with both the original and generated causes are in terms of readability.

We leveraged a standard Python library [223] to analyze the grammatical correctness of sentences with both the original and generated causes, and it identified a total of 6 and 7 grammatical issues respectively for a total of 10 test data-points. We have also measured the GRUEN score [219] which represents the general linguistical quality of texts. Our test sentences with generated causes achieved an average GRUEN score of 0.7421 with a standard deviation of 0.1432, while the original text scored 0.7473 and 0.1368 for the same metrics respectively.

Metric	Text with the original cause	Text with generated cause
Flesch Reading Ease [217]	63.19	70.84

Coleman Liau Index [218]	9.63	8.06
Grammatical Issues Count [223]	6	7
GRUEN [219]	0.7473	0.7421

Table 15: Evaluating the linguistic quality of the generated causes using methods that do not use original text as a reference

4.8.3.3 Sentiment and Emotion Analysis

We perform sentiment and emotion analysis of the test data points with original and the generated causes. Findings from these analysis is presented in Table 16. We observe that for all the 10 data points, emotion labels from the dataset, emotion labels from the emotion classification algorithm for the test data points with original and generated causes agree with each other except only for data point #6, where the emotion label from the dataset is "Surprise" but classified by the algorithm used (i.e., "emotion-english-distilroberta-base" model [221]) as "Fear" for both the test data points with the original and the generated causes. From the sentiment analysis results (using "twitter-roberta-base-sentiment" [220]), we notice that all the test data points with original and generated causes agree with each other except for the data point #7, which for the test data with original is classified as "Neutral", whereas for the generated one, it is classified as "Negative".

#	**Emotion** (from dataset)	**Emotion** (Original)	**Emotion** (Generated)	**Sentiment** (Original)	**Sentiment** (Generated)
1	Anger	Anger	Anger	Neutral	Neutral
2	Fear	Fear	Fear	Neutral	Neutral
3	Sad / Sadness*	Sadness	Sadness	Negative	Negative
4	Fear	Fear	Fear	Negative	Negative
5	Anger	Anger	Anger	Negative	Negative
6	Surprise^	Fear^	Fear^	Negative	Negative
7	Sad / Sadness*	Sadness	Sadness	Negative	Negative
8	Fear	Fear	Fear	Neutral^	Negative^
9	Happy / Joy*	Joy	Joy	Positive	Positive
10	Happy / Joy*	Joy	Joy	Neutral	Neutral

Table 16: Sentiment and emotion analysis of the test data point with original and generated causes. The rows follow the same order as Table 13, and the serial numbers in left most column correspond to the serial numbers in the left most column of Table 13. Asterisk (*) indicates synonymous emotion label added for the ease of manual comparison. Circumflex (^) indicates mismatch.

4.9 Discussion

Based on our observation, we opine that the generated causes for our test data are in general representative of the emotions expressed in the respective sentences. We observe that a few

generated causes seem somewhat “bizarre”, however, they still seem to be relevant for the emotion expressed in the sentence.

It is also noteworthy that the generated causes generally maintained grammatical correctness. It is also confirmed by the comparative GRUEN score [219], followed by other quantitative analyses such as Coleman Liau Index [218], and Grammatical Issues Count [223], that the grammatical and semantical correctness of the sentences with the generated causes is similar to the sentences with the original causes. The GRUEN score achieved in the text with generated causes also indicates the low redundancy and high focus and coherence of the generated causes [219].

We also evaluated our generated causes using BLEU [213], METEOR [214], ROUGE [214], and BERTSCORE [216]. These techniques typically attempt to predict the “similarity” of the generated text with some reference text. These techniques, while typically used for machine learning, have also been used to evaluate other text generation tasks due to their generalizability [222]. Our test sentences with generated causes achieved a decent score for these metrics. For instance, a BLEU score of more than 0.50 is interpreted as “very high-quality translation” for the machine translation task [213]. Our achieved scores for these metrics, which are all well above 0.50, indicate the decent quality of the generated causes with reference to the original causes.

Overall, based on our observation and linguistic quality evaluations by techniques like GRUEN [219], and other scores such as BLEU that indicate semantic similarity of the generated causes with the original ones, we assert that the generated causes using the “infilling” method meaningfully represent the emotion expressed in the given text while acknowledging that there is room for improvement.

Based on the sentiment and emotion analysis results presented in Table 16, we observe that the emotion labels of the test data with the original and generated causes, as well as the emotion labels from the dataset, agree with each other for 9 out of 10 data points. Furthermore, the emotion labels for the test data with the original causes and generated causes agree with each other 100% of the time. In terms of sentiment analysis, the sentiment of the test data with the original causes matches the sentiment of the test data with the generated causes for 9 out of 10 data points. These results suggests that the generated causes are meaningful and relevant to the emotion expressed in the sentence.

We also highlight the need for large-scale human evaluation of the generated causes when evaluating the performance of techniques to perform the proposed ECG task.

In summary, we believe that this research work demonstrates the viability of the proposed ECG task, particularly as an "infilling" text generation task. We anticipate that this work will serve as the foundation for further research focusing on developing novel techniques to perform the ECG task.

4.10 Limitation and Future Work

The focus of this study is to introduce the new task of "Emotion Cause Generation" (ECG) and demonstrate the viability of the task through technical implementation. The technical implementation itself is not the primary focus of the study. There is a vast opportunity to develop novel technical solutions with regard to the proposed ECG task. We plan to navigate these areas in our future studies.

Apart of individual human differences, there can be many common collective factors that can have impact on the cause of an expressed emotion, such as culture, geography, political and economic status of a region etc. For example, people living in regions where snowfall is common can associate many emotions with snow, which may not be the case for people living regions where snowfall never happens. In this study, we present ECG in a generic and simplistic manner, and we modified an existing infilling mechanism to demonstrate how ECG can be performed in general as an infilling task. There are opportunities to introduce more sophistications to the ECG task by bringing in factors such as cultural context in the cause generation process. For example, we can further train a machine learning model that can perform the ECG task with awareness of a specific culture by infusing it with specific cultural information. We believe these are interesting avenues to explore in future studies related to the ECG task.

The evaluation of the technical solution used in this study to perform the proposed ECG task was done on a limited dataset of 10 generated causes containing texts. This was due to the scope of the study, which was primarily to introduce the ECG task along with assessing its viability as an NLP task and not to propose a novel machine learning architecture to perform the proposed task.

We perform a manual inspection of the generated causes to understand their meaningfulness coupled with several quantitative analyses. This manual inspection by the researchers would not

have been possible for a large number of datapoints. In future, particularly for studies where novel technical solutions for the ECG task are proposed, we recommend performing analysis on more test datapoints and conducting human-evaluations of the generated texts through distributed user surveys.

In addition to human-evaluations, which can be tedious and difficult to scale, a worthwhile investigation can be attempting to find and / or develop automatic evaluation metrics for the ECG task. While the metrics for general linguistic quality is useful, they are not capable for evaluating the relevancy or meaningfulness of the generated causes. The other automatic metrics we used such as BLEU are typically used for more structured generative tasks such as machine translation where a finite set of correct answers exists. However, for open-ended tasks such as the proposed ECG task, the correct answers for a single data point can be numerous. As a result, metrics such as BLEU has less utility for the ECG task. We believe these indicate the need for more automatic evaluation methods for open-ended generative tasks such as the one proposed in this chapter.

4.11 Conclusion

In this study, we introduce the ECG task, the first generative NLP task in the ECA domain that aims to generate meaningful cause-spans for the emotion expressed in a given text. We customize an existing text-infilling mechanism and fine-tune it further to demonstrate the viability of the ECG task. We observe that the fine-tuned model is able to generate reasonably meaningful causes for most of the example inputs. We emphasize that the focus of this work is to introduce the new ECG task as a viable NLP task while acknowledging the limitations in our technical demonstration and evaluation. We hope that this work will inspire researcher communities to develop novel techniques to perform the proposed ECG task.

Chapter 5: ECSGen and iZen: A New NLP Task and a Zero-shot Framework to Perform It

Building upon the previous chapter where we introduced the first-ever generative NLP task in the Emotion-Cause Analyses (ECA) domain, in this chapter, we introduce another novel generative task within the ECA domain named, Emotion-Cause mitigating Suggestion Generation (ECSGen). The objective of this task is to generate pertinent suggestions to alleviate the underlying reason for a negative emotion conveyed in a given text. We also propose iZen, a technical framework to perform this task in a zero-shot manner, without requiring any new training or fine-tuning steps. We curate new datasets to evaluate iZen's ability to perform the ECSGen task. Our rigorous experiments and analysis, which included human evaluations and some complementary automatic metrics, suggest that iZen is capable of generating relevant suggestions as part of the ECSGen task.

The work presented in this chapter has been submitted to one journal[4].

5.1 Introduction

Machine learning algorithms process a large amount of data to generate machine learning models. These models then can be used to perform the intended tasks, such as Sentiment Classification, translation, and face recognition. Generally, machine learning algorithms learn by optimizing loss function (i.e., objective function) [224]. This optimization takes place by iterating through a vast amount of relevant data (i.e., training data), and adjusting the internal model parameters accordingly. [1]

There are many different algorithms to train machine learning models. Some common among them include, but are not limited to, Linear Regression, Logistic Regression, Random Forest Algorithm, K-Nearest Neighbor (K-NN) Algorithm, and different variants of Artificial Neural Network (Convolution Neural Network, Recurrent Neural Network, Long Short-Term Memory, etc.) [225], and Transformer (which introduces the notion of self-attention [95]). Transformer enabled the

[4] [IN REVIEW] Riyadh, M., & Shafiq, M. O. (2023). ECSGen and iZen: A New NLP Task and A Zero-shot Framework to Perform It.

rapid development of large pre-trained language models (LM). Over the past few years, many such large-scale pretrained models have been released, typically containing billions of parameters. Some notable examples include Bidirectional Encoder Representations from Transformers (BERT) [120], Generative Pre-trained Transformer (GPT) [127] etc. Facebook recently released one of the most advanced open-source pretrained LMs, named Open Pretrained Transformer (OPT), which has 175 billion parameters in its largest variant [130]. These LMs can generate meaningful texts based on the prompt provided without requiring any further training. A process named fine-tuning [120] is typically performed to use these models in specific downstream tasks such as question answering, summarization, and sentiment analysis [226]–[230]. This fine-tuning process provides a way to control text generation for different outcomes [128]. Although fine-tuning process leverages the advantages of pre-trained models, it is still a training process that requires curated datasets and compute-resources.

Apart from fine-tuning, another method to control text generation from LMs is tuning the prompt provided to it in order to elicit the desired response. Recently, many researchers have focused on developing prompt engineering techniques [209], [231]. We believe that the prompt engineering techniques and their possible applications within NLP can benefit from further investigation, particularly in designing plausible novel NLP tasks and related techniques to perform those tasks. As part of this study, we are interested in exploring this area further, particularly within Emotion-Cause Analysis (ECA) [232], which is a specific domain within the NLP field.

In recent years, the ECA research area has flourished significantly [90], [194], [207], [233], [234]. However, these studies are largely focused on classification tasks, with little to no emphasis on the utility of LMs and their text generation capability. We introduced the first generative ECA task, named Emotion Cause Generation (ECG) [see Chapter 4]. In ECG, we attempted to generate a relevant cause-span for the given emotion expressed in a sentence. We positioned the ECG task as an "infilling" task [202], [235], where the cause-span was masked. We customized an existing LM-based infilling technique [202] to generate a relevant cause-span in place of the mask.

Building upon these studies, particularly the introduction of ECG [see Chapter 4], the potential utility of LMs in ECA related research, and various techniques to use these LMs without any new training steps, we propose a new ECA-related generative NLP task, named ECSGen (**E**motion-

Cause Mitigating **S**uggestion **Gen**eration) and a zero-shot [236] framework, named iZen to perform this task.

5.2 Contributions

The specific contributions of the research work described in this chapter include the following:

- We propose a novel generative NLP task within the ECA domain named Emotion-Cause mitigating Suggestion Generation (ECSGen) that aims to generate suggestions to alleviate the underlying reason for a negative emotion expressed in a given text.
- We devise a framework named iZen that can perform this task in a zero-shot manner, i.e., requires no specific training or labeled data to perform this new task.
- We create our own datasets containing input statements for iZen to perform the ECSGen task. We make one of our curated datasets publicly available, which contains 200 input statements, 6 generated suggestions by iZen for each of these input statements (a total of 1200 suggestions), along with the relevancy score (provided by human participants) for each suggestion and information on contextual richness of the input statements. We expect this dataset to encourage further research in this area.
- We perform rigorous experiments along with human evaluation in order to understand iZen's capability to perform the proposed ECSGen task.

5.3 Related Studies

Rapid advancement in machine learning technologies is having a significant impact on numerous areas. NLP is one of the prime examples of such areas. A relatively recent addition to machine learning technology is Transformer [95], which acted as a catalyst to develop more sophisticated machine learning models, such as large language models (LM) [128]. These LMs and their unique capabilities inspire a plethora of new research studies in NLP. For example, Dathathri et al. [237] introduced Plug and Play Language Model (PPLM) which incorporates an LM with one or more classifiers to guide the text generation without requiring any additional training or fine-tuning. Following this work, several other Attribute-based Controlled Text Generation (CTG) [238] related techniques emerged [239]–[241], such as Future Discriminators for Generation (FUDGE)

[242], which only requires access to the logits of the language model output for imposing a variety of controls (e.g., formality) on the text generation.

LMs such as OPT [130] have significant reliance on the "prompt" (i.e., an initial snippet of text based on which LMs generate more text) in terms of deciding what to generate. This led many researchers to investigate designing prompts in various ways in order to get different desired outcomes from the LMs. For instance, Yang et al. [243] introduced Text-attribute Controller (Tailor) which is a prompt-based approach to control text generation. They propose mechanisms to control text generation with single and multiple attributes. However, their multi-attribute text generation required training a separate component that converted two single attributes to a multi-attribute controller. In addition, their method was also limited to two attributes, leaving room for further exploration in this area. These approaches are also sometimes referred to as "prompt tuning". Like Tailor [243], these approaches may involve some sort of training in order to generate the appropriate prompt. Many other examples of studies with a similar goal exist in the literature [209], [231], [244], [245]. The main intuition behind this type of prompt tuning is to trigger a desired response from the LM with task relevant prompt parameters without needing to update any parameters in the original LM.

One of the goals of this study is to explore the concept of prompt tuning in the domain of ECA [232], which is a specific research area within NLP. Within ECA, several different tasks exist today. For example, Emotion-Cause Extraction (ECE), Emotion-Cause Pair Extraction (ECPE), and Emotion Cause-Pair Span Extraction (ECSP) [[76], [92], [194], [206], [246]].

Lee et al. [77] proposed the ECE task, which was the first task within the ECA domain. The challenge was to identify the reason for a particular emotion that was represented in a paragraph. Further investigations, including those utilizing rule-based and conventional machine learning technologies, were conducted after this introductory work ([9], [203], [205]). Soon after, more sophisticated deep-learning based methods for performing ECA tasks appeared. These included a technique developed by Gui et al. [87], which leverages a question-answering model and a deep memory network to determine the source of emotion. Another example was the usage of a Convolutional Neural Network in accomplishing the ECA task [204].

The design of more ECA related innovative tasks followed in the subsequent years. For example, Xia et al. [90] introduced ECPE, which classifies emotion in a text and detects the cause of that

emotion. Leveraging BERT, Fan et al. [207] reconstructed the ECPE problem as a method of directed graph creation. These ECA task variations identified emotion-cause at a clause-level. An issue with this was that these clauses often contained information irrelevant to the cause of the emotion. Li et al. [194] attempted to address this issue with their proposed ECSP task that extracts the span of the cause from a given sentence instead of detecting the cause-clause from multiple options. We introduced the first generative task within ECA, named ECG [see Chapter 4], which we discussed in the previous chapter. For ECG, the task is to generate a relevant cause of the emotion expressed in an input statement, where the "cause-span" is masked in that statement. This study is largely inspired by the ECG task and builds upon it to propose another novel generative NLP task within the ECA domain, named ECSGen.

5.4 ECSGen: The Proposed Task

In this study, we propose a new ECA-related generative NLP task, named ECSGen (**E**motion-**C**ause Mitigating **S**uggestion **Gen**eration). It takes a given sentence that expresses a negative emotion (e.g., sadness, frustration) and an underlying cause of that emotion as input and aims to generate relevant suggestion(s) (e.g., an activity or idea) to mitigate that emotion. For instance, if the input statement is - "*I am bored because I don't enjoy this game anymore*", then an expected relevant suggestion could be - "*Maybe try a new game!".* Figure 26 demonstrates this example. In this instance, the emotion expressed is "bored", the causal conjunction used is "because", and the cause of the emotion is, "I don't enjoy this game anymore".

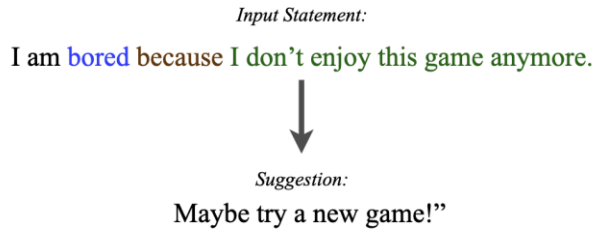

Figure 26: An example of the ECSGen task. The input statement expresses an emotion with an associated cause and the ECSGen task requires generating relevant suggestion(s) to mitigate the cause behind the emotion

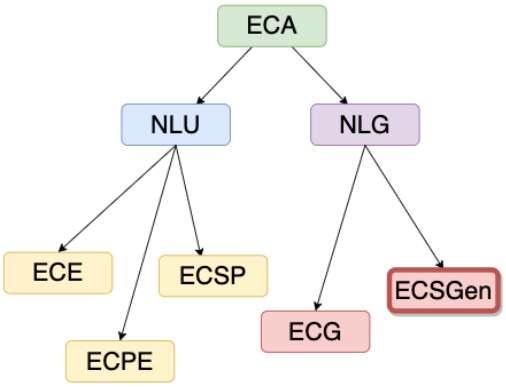

Figure 27: ECSGen and other ECA tasks

Figure 27 demonstrates how ECSGen fits into the ECA domain among other NLP tasks. ECSGen is the second NLG task in this domain.

5.5 iZen: A Framework to Perform ECSGen

In this study, we propose a framework, named iZen (i.e., the name iZen does not possess any specific meaning), to perform the proposed ECSGen task. We first describe the scope of iZen in relation to the proposed task, and then we describe its technical architecture.

5.5.1 Scope of the Proposed Framework:

iZen expects input with certain constraints. The desired input text is a simple sentence where a person expresses his/her emotion and states a cause behind that emotion. iZen requires the input to be formulated in the following specific pattern:

I am <express emotion> <causal conjunction> <cause>

The following example demonstrates the above pattern:

I am <feeling sad> <because> <I failed the exam>

The pattern is broken down below with further details outlining the required input constraints:

1. The input text starts with "singular first person" (i.e., "*I*"). In the input text, an emotion is expected to be described in "singular first person". For example, "*I am feeling sad*", "*I am bored*", "*I feel frustrated*" etc.

2. Immediately after the emotion is expressed, it is followed by a causal conjunction such as "*because*", "*because of the fact that*" etc.
3. After the causal conjunction, the cause of the emotion is stated. In the above example, "I failed the exam" is the cause behind the emotion expressed in that sentence.

Points 1 and 2 from the above list together form the "beginning part" (expression of emotion and causal conjunction) of the input text. In this study, the following "beginning parts" were used:

- *"I feel disgusted because",*
- *"I feel down because",*
- *"I feel bored because",*
- *"I am bored because",*
- *"I am feeling sad because",*
- *"I am annoyed because",*
- *"I am frustrated because",*
- *"I feel sad because of",*
- *"I am angry because of the fact that",*
- *"I feel angry because"*

The negative emotions used in the "beginning parts" included, "*sadness*", "*anger*", "*disgust*", "*annoyance*", "*frustration*" etc. We selected these emotions from the collection of various types of emotions in Plutchik's Wheel of Emotion [11].

The zero-shot nature of iZen obviates the need for additional training, fine-tuning, and hyperparameter tuning. The selection of off-the-shelf (OTS) components within the iZen architecture was based on a variety of criteria, including their domain usage, reported superior performance in specific tasks, and availability in a suitable Python package. For instance, the widely used "twitter-roberta-base-sentiment" model [220] from the Cardiff NLP research group [247], which satisfies most of these criteria, was utilized. Furthermore, before final integration within the architecture, we evaluated the suitability of the selected OTS components for the specific task segment in iZen through experimentation during the development phase.

5.5.2 iZen's Architecture:

iZen is comprised of 4 main components:

- PromptGenie *(P)*
- Zen *(Z)*
- Eliminator *(E)*
- Guardrail *(G)*

The adjacent letters represent the symbols used to identify these components in this study. Each of these components is described below.

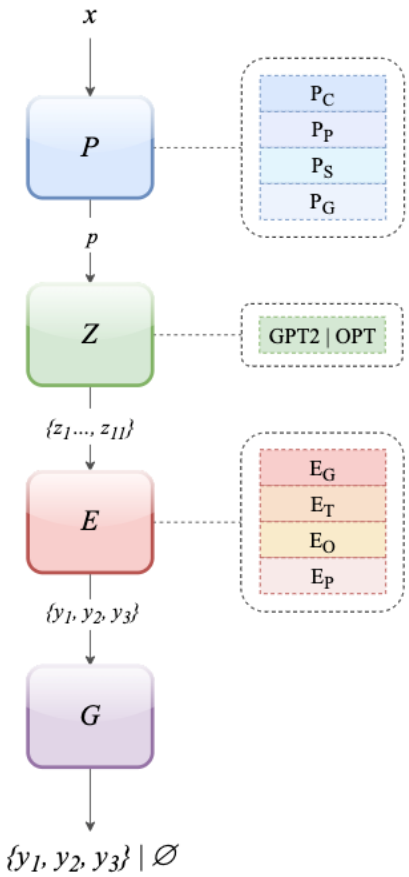

Figure 28: iZen Framework's Architecture: PromptGenie, Zen, Eliminator, and Guardrail are denoted with P, Z, E, G (respectively). P_C refers to PromptGenie's Cause-span Extractor, P_P refers to Prompt Prefix Generator, P_S refers to Cause-span Simplifier, and P_G refers to Grammar Corrector. E_G, E_T, E_O, E_P refer to Eliminator's calculators for Grammatical Correctness, Toxicity, Optimism, Positivity respectively. Outputs from PromptGenie, Zen, and Eliminator are indicated with p, z, and y respectively. Guardrail outputs are denoted with y or ∅ (null or no suitable suggestion generated).

5.5.2.1 PromptGenie (P)

PromptGenie is predominantly a rule-based prompt generator for ECSGen. Given an input in the expected format described in the previous section, PromptGenie generates a prompt that gets supplied to the next component in the pipeline in iZen: Zen. The generated prompt is significantly important for the eventual outcome of iZen since it is the input used by Zen that generates the candidates for desired suggestions.

As described in the previous section, the expected input (x) has the following format:

I am <express emotion> <causal conjunction> <cause>

The purpose of PromptGenie is to take the "cause" part of the original input and leverage it to form a text in a manner that catalyzes a reversal of the cause. Given an original input statement (x), PromptGenie generates a prompt (p) in the following pattern:

If you <want | don't want> <text span crafted based on the cause-span in x>

To generate the desired prompt (p) from the given input statement (x), PromptGenie goes through these four main steps:

1. Extract cause-span
2. Generate prompt prefix
3. Simplify the identified cause-span
4. Grammar correction

These four operations are supported by PromptGenie's four different sub-components (in the same order as the steps mentioned above):

1. Cause-span Extractor (P_C)
2. Prompt Prefix Generator (P_P)
3. Cause-span Simplifier (P_S)
4. Grammar Corrector (P_G)

To elaborate on PromptGenie's process, we can consider the same example in the previous section:

I am <feeling sad> <because> <I failed the exam>

Here, the cause part is the following: "*I failed the exam*". PromptGenie's Cause-span Extractor (P_C) first extracts this cause part (p_c) based on the rules developed around the "causal conjunction" set used in this study (i.e., as mentioned in the previous section: "*because*", "*because of*" etc.).

After the cause is identified and separated, PromptGenie's Prompt Prefix Generator (P_P) generates a prefix (p_p) to be added to the cause. The prefix is determined based on the cause. For example, for the cause "*I failed the exam*", the generated prefix would be, "*If you don't want to*".

The extracted cause (p_c) is further simplified (p_s) by the Cause-span Simplifier (P_S) before the prefix is added to it. For instance, it is transformed to present tense, and the subject is stripped out (e.g., "*I*"). The example cause then becomes: "*fail the exam*". After this, PromptGenie adds a prefix to this simplified version of the cause: "*If you don't want to fail the exam*". PromptGenie also includes logic to handle cases where the original cause contains a negation. For example, if the input is the following:

"I am sad because I did not pass the exam."

Then, PromptGenie generates the following prompt:

"If you want to pass the exam".

Finally, PromptGenie's Grammar Corrector (P_G) leverages an off-the-shelf (OTS) pretrained grammar corrector model [248] in order to rectify grammatical errors that can be caused during different steps of rule-based transformations in the prior steps mentioned.

5.5.2.2 Zen (Z)

Zen is the main generator component of iZen that generates the candidates (z_i, where $i \in \{1...11\}$) for the desired suggestions based on the prompt (p) supplied by PromptGenie. Zen is primarily a wrapper around the pretrained language model (LM) of choice. In this study, we included two LM options for Zen, – OPT [130] and GPT2 [128]. We chose OPT as it is one of the latest and most advanced general purpose open-source LM available publicly. Particularly, the "opt-1.3b" [249] (i.e., contains 1.3 billion parameters) variant of OPT was used as larger variants with more parameters required significantly more computing resources than the resources that were available to us (i.e., Google's Colab Pro [250]). GPT2 is chosen due to its significant adoption across NLP research as well as the fact that it is open source and researchers have unrestricted access to it,

unlike some other LMs such as GPT3 [129]. We used the "gpt2-xl" variant [251] as it is the closest to our OPT variant of choice in terms of the number of parameters (i.e., 1.5 billion).

Within Zen, we leverage many of the default configurations [252], [253] for OPT and GPT2 models with some customizations to suit the specific needs of this study. For instance, we include a custom "Stopping Criteria" function within Zen that dictates when it must stop generation. We use some characters to signal Zen to stop text generation, which includes, full stop or period ("."), new line character ("\n"), exclamation mark ("!"), and semi-colon (";"). Zen sets a few other configurations for the LMs in order to suit the purpose of the study. This included setting the maximum number of new tokens generated by the LMs to 100, and setting the minimum generation length to 30 characters. Since our goal was to generate short and simple suggestions, we did not want our suggestions to be more than one sentence long, which is supported by our custom "Stopping Criteria" functions along with these LM configurations. The minimum length condition was used to ensure there is sufficient room for meaningful text. In addition, we leveraged "multinomial sampling" as our decoding strategy [254], which allows for generating different outputs for the same input in each inference. For each input, Zen generated 11 candidate suggestions (the choice of this number is explained in the next section), which are then fed into the next component in iZen's pipeline: Eliminator.

5.5.2.3 Eliminator (E)

Eliminator sits next to Zen in iZen's architecture where it takes in 11 generated suggestions (z_i, where $i \in \{1…11\}$) from Zen against one input statement (x). This component is a hybrid of rule-based and machine-learning based mechanisms. It is built on top of some OTS pretrained models and libraries. The goal of this component is to find the top 3 suggestions (y_1, y_2, y_3) out of the 11. The elimination of the rest of the suggestions occurs based on certain elimination criteria: grammatical correctness, optimism, positivity (i.e., positive sentiment), and toxicity. The selection of these criteria is mainly inspired by a relevant prior study [255]. In addition, we use "toxicity" as a criterion as LMs can generate toxic content depending on the prompt [129]. Also, noteworthy that with the intent to make Eliminator more robust, we used optimism and positivity as two separate criteria, filtered using two separate OTS pretrained models, though they may sound similar.

Eliminator operates on two primary steps: (1) Calculate candidate suggestions' scores for different elimination criteria, (2) Identify top-k (for this study, k is 3) suggestions among the generated candidate suggestions based on these scores. Step 1 accommodates different sub-steps. As mentioned above, for this study, these sub-steps include the following:

a. Calculate grammatical correctness score (*g_s*)
b. Calculate toxicity score (*t_s*)
c. Calculate optimism score (*o_s*)
d. Calculate positivity score (*p_s*)

These four steps are supported by four different sub-components of Eliminator, which are based on three separate OTS pretrained models and a Python library. These are listed below in the same order as the steps:

a. Grammatical Correctness Calculator (E_G): It is based on the "language-tool-python" package [256]
b. Toxicity Calculator (E_T): This component is based on the "toxic-bert" pretrained model [256]
c. Optimism Calculator (E_O): This component leverages the "twitter-roberta-base-emotion" pretrained model [257]
d. Positivity Calculator (E_P): It is based on the "distilbert-base-uncased-finetuned-sst-2-english" pretrained model [258]

After these subcomponents calculate the elimination criteria scores (e_{scores}), the second main step of the Eliminator takes place. This step is carried out by the fifth and final subcomponent of the Eliminator, named Selective Deduction Unit (E_{SD}). E_{SD} algorithm eliminates the two least favorable suggestions for each criterion in the following order:

- First, it eliminates the two suggestions out of 11 that contain the most grammatical errors. If there are suggestions with the same number of grammatical errors, then two among them are chosen randomly. At the end of this step, nine suggestions are remaining.
- In the second step, Eliminator removes the two most toxic suggestions from the remaining nine suggestions, leaving seven suggestions for further filtering.
- Third, Eliminator removes the two least optimistic suggestions from the seven. At the end of this step, five more suggestions are remaining.

- Lastly, Eliminator discards the two least positive suggestions from the five supplied in the previous step. Finally, this leaves us with the top 3 suggestions.

The E_{SD} algorithm was designed to function on a "deduction unit" at each step. For this study, this deduction unit was two and we had four steps (or criteria). As a result, 11 outputs get reduced by two at each step and eventually resulting in the top 3 outputs.

As described above, E_{SD} follows a greedy algorithm [259] approach (i.e., makes locally optimal choices at each step with the hope of finding a global optimum) in a specific order: grammatical correctness, toxicity, optimism, positivity. The reason for this is to remove the grammatically incorrect and potentially toxic suggestions first as they are critical to remove. The last two criteria further filter the rest of the suggestions to eventually select the final three suggestions. However, it is worth mentioning that this can be achieved in a few other apparent ways instead of using greedy algorithm approach. For example, we can sum all the scores from the Eliminator subcomponents using appropriate weights (e.g., if higher weight indicates more likely candidate for removal, then toxicity and grammatical correctness should have more weights than the other criteria) in order to select the top three suggestions.

The top 3 suggestions selected by E_{SD} then can either be chosen as the final output or can be forwarded to the next but optional component of iZen: Guardrail, which we describe below.

5.5.2.4 Guardrail (G)

Guardrail is an optional safeguard component that aims to automatically discard an entire output if there is any serious concern identified in any part of the top 3 suggestions. iZen's Guardrail component checks for profanity in the top 3 suggestions supplied by Eliminator, leveraging a Python library named "profanity-filter" [260]. If there is any profanity identified by Guardrail in any of the top 3 suggestions for an input statement, then the entire output is discarded. This means, Guardrail either outputs the same top 3 suggestions (y_1, y_2, y_3) supplied by the Eliminator if they qualify as the final output of iZen for a given input statement (x), otherwise it outputs ∅ (null or no suggestions).

In this study, we opted to use this component in order to further safeguard against unintended outputs. Eliminator, with its filtering based on toxicity, positivity, and optimism, provides a robust mechanism to filter out unintended texts. However, based on our observation, even with such

robust filtering, there were still some suggestions among the top 3 that included profanity. Though this represents a negligible amount of the generated suggestions in our experiment (about 1.6%), nonetheless, we opted to include this final Guardrail component as an additional safety net in order to ensure, as much as possible, that iZen's outputs do not include any profanity. For this first release of iZen, we chose to discard all top 3 suggestions when profanity was identified by the Guardrail component in any of the suggestions. This is to maintain iZen's input/output consistency (i.e., three suggestions for one input statement).

The reason we chose to have the Guardrail as a separate component and not as a part of the Eliminator is because of the distinct goals of these two components. Eliminator essentially ranks the generated suggestions and selects the top 3 among them based on some elimination criteria. Whereas Guardrail's criteria (in our case, it is profanity; however, we expect this to be extensible to other factors in the future) are such that even if there is a little profanity for example, it may need to be excluded – it is intentionally more binary and stricter in nature. As a result, we believe Guardrail serves best as a separate last optional checkpoint instead of being combined with the Eliminator component which is more akin to ranking systems.

5.5.2.5 iZen's Overall Algorithm

As discussed in detail in the previous section, iZen takes in an input statement, x, and attempts to generate three relevant suggestions, y_1, y_2, y_3. Algorithm 3 summarizes iZen's overall process to generate suggestions for each input as described in the previous section:

- First, PromptGenie's (P) Cause-span Extractor (P_C) subcomponent takes in the input statement (x), and then it goes through the other three subcomponents of PromptGenie in series. *p* is the final output from PromptGenie.
- Zen (Z) generates 11 candidate suggestions: $\{z_1 ..., z_{11}\}$
- Next, the four calculators of the Eliminator component (E_G, E_T, E_O, E_P refer to Eliminator's calculators for Grammatical Correctness, Toxicity, Optimism, Positivity respectively) calculates the respective scores for each of the generated suggestions and appends them in e_{scores}
- The fifth and the final subcomponent of the Eliminator, named Selective Deduction Unit (E_{SD}) takes in $\{z_1 ..., z_{11}\}$ and the associated e_{scores} and selects the top three generated suggestions: $\{y_1, y_2, y_3\}$

- The last safeguard component, Guardrail (G) checks for profanity in $\{y_1, y_2, y_3\}$ and passes them as the final output of the iZen architecture if no profanity if found. In case any profanity is found, it outputs null ($\emptyset$)

Algorithm 3: iZen's Overall Algorithm. Meaning of symbols used: PromptGenie, Zen, Eliminator, and Guardrail are denoted with P, Z, E, G (respectively). P_C refers to PromptGenie's Cause-span Extractor, P_P refers to Prompt Prefix Generator, P_S refers to Cause-span Simplifier, and P_G refers to Grammar Corrector. E_G, E_T, E_O, E_P refer to Eliminator's calculators for Grammatical Correctness, Toxicity, Optimism, Positivity respectively. Outputs from PromptGenie, Zen, and Eliminator are indicated with p, z, and y respectively. Guardrail (G) outputs are denoted with y or ∅ (null or no suitable suggestion generated).

```
Input(s): x
Output(s): {y1, y2, y3} or ∅
// PromptGenie steps
pc ← PC(x)
pp ← PP(pc)
ps ← PS(pc)
pd ← pp + ps
p ← PG(pd)

// Zen step
{z1…, z11} ←Z(p)

// Eliminator steps
declare escores: collection of scores from Eliminator subcomponents

for z ∈ {z1…, z11} do
      g_s ← EG(z)
      t_s  ← ET(z)
      o_s ← EO(z)
      p_s ← EP(z)
      append {g_s, t_s, o_s, p_s} in escores
end for

ESD(z1:11, escores) ← {y1, y2, y3}

// Optional Guardrail step
{y1, y2, y3} or ∅ ← G(y1:3)

return: {y1, y2, y3} or ∅
```

5.6 Experiments and Evaluation

5.6.1 Procedure

5.6.1.1 Suggestion Generation

The initial steps in setting up our experiments involved developing iZen's pipeline (see section 5.5 for more details). This pipeline was then used to generate suggestions for inputs contained within our curated datasets (see sub-section 5.6.3 below). Our experiments included four variations in terms of input dataset and LM used in iZen's Zen component. We used two input datasets, each

containing 200 input datapoints for iZen. This sums up to 400 unique input datapoints. As LMs, we used OPT and GPT2 (see section 5.5.2.2 for more detail). Therefore, iZen with OPT generated three suggestions for each of the 400 datapoints, and iZen with GPT2 generated three suggestions for each of the 400 datapoints. This sums up to 800 input datapoints for iZen, and for each of them, three suggestions were generated, summing up to 2400 suggestions. Figure 29 visually summarizes this for further clarity.

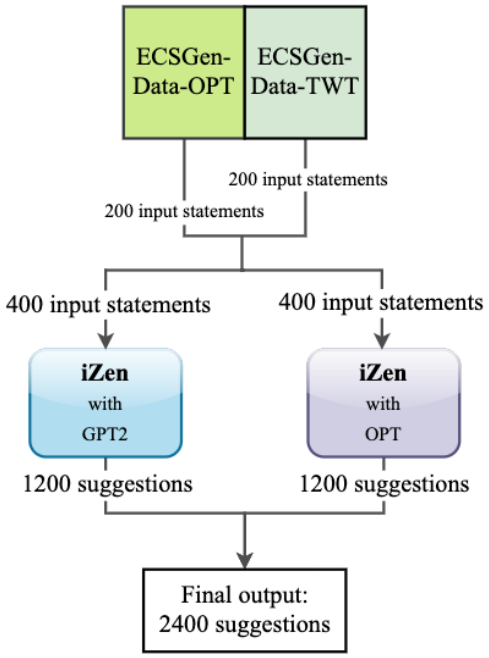

Figure 29: An overview of how 800 input statements (400 unique) resulted in the final 2400 suggestions in this study.

As mentioned in the earlier sections, we opted to use iZen's optional Guardrail component in order to further ensure that iZen's outputs are free of profanity. In our experimental setup, after the Eliminator component selects the top 3 suggestions, it goes through the final "Guardrail test" (i.e., filtered based on Guardrail's configuration). After the top 3 suggestions "pass" the "Guardrail test" (representing about 98.4% of the suggestions in our experiments), they are accepted as the final outputs of iZen. Since the suggestions that "failed" the "Guardrail test" are negligible (only 1.6% of the generated suggestions), we excluded them from our analysis for simplification. Consequently, the results reported in the subsequent sections are based on 98.6% of the datapoints (i.e., this represents the 800 input statements mentioned above), and the remaining 1.6% is

considered as the overall error margin. We opted to accept this error margin in our reported result due to its insignificance and in exchange for the additional safeguard it provided for our generated outputs which we present to our human evaluators.

5.6.1.2 Evaluation of the suggestions

Our goal with iZen is to generate simple and relevant suggestions to mitigate the emotion-causes expressed in the input texts. iZen's internal mechanisms ensure that the generated texts are simple (i.e., keeping the text short, one sentence long). However, in order to "measure" relevancy (i.e., how relevant generated suggestions are in relation to the input statement containing an emotion with an associated cause), the suggestions are required to be evaluated by humans. We consider this as the primary method for evaluating the generated suggestions for their expected quality in this study. In addition, we also leverage some automatic metrics as complementary measurements to evaluate the generated suggestions. The following sections include details regarding both types of evaluation.

5.6.1.3 Human Evaluation

Data Collection: We conducted a survey with 20 participants, aged between 18-50 years. All our participants were required to be able to read and understand English. We designed the survey using Qualtrics [261] and distributed it online using Amazon MTurk [262]. We leveraged some of the options provided by these tools to ensure the integrity of the survey responses. For instance, we required our survey participants to have certain MTurk System Qualification Scores (e.g., "Number of HITs Approved": above 10,000 and HIT "Approval Rate": above 90%) in order to ensure their reliability. We also examined the survey responses manually in order to identify outliers based on criteria such as completion time (i.e., if a participant completed the survey extremely faster compared to the other participants), and sanity check of the responses (i.e., if many of the responses seemed nonsensical based on our observation. For example, if many of the obvious completely irrelevant suggestions were marked as "*Very relevant*" and vice versa). We released our survey in several batches through MTurk until we received 20 valid sets of responses (i.e., a total of 26 participants completed our survey, and among them, 6 were discarded based on the criteria mentioned above).

The survey contained the 800 final datapoints used in iZen, including 800 input texts and three output suggestions for each. As mentioned above, these datapoints can be categorized into four groups based on the input dataset used for generation and LM used within iZen. Consequently, each of these four groups contained 200 input datapoints.

Each participant in our study was presented with an equal number of questions (i.e., 10) from each of the four categories, which summed up to 40 datapoints per participant. Each of these 40 datapoints constituted a question set. A question set included the original input statement that contained an emotion and an associated cause and the three generated suggestions from iZen. We asked the participants to rank the relevancy of each suggestion in relation to the original input statement using a 5-point Likert scale. In order of relevancy (high to low), our Likert scale labels were: "*Very relevant*", "*Moderately relevant*", "*Somewhat relevant*", "*Slightly relevant*", and "*Not at all relevant*" (label names are inspired by prior studies and guidelines [263], [264]). We also asked them to share their opinion on whether the original text contained sufficient contextual information in order to provide a relevant suggestion. At the end of the survey, the participants were also presented with two open-ended questions to share their opinion on their overall thought about the suggestions (i.e., what they liked and disliked about the suggestions). We included some sample questions in the Appendix to demonstrate the overall structure of the survey.

We have attained ethics clearance from Carleton University Research Ethics Board-B (CUREB-B)[5] to perform this evaluation with human participants.

Data Analysis: To analyze our collected data, primarily Likert scale responses which represent our intended measurements, we report the percentages of different labels of our Likert scale. For example, we report how many of our survey participants (in percentage) found at least one of the top three suggestions "somewhat or more relevant" for each question-set, or how many of the participants thought all of the three generated suggestions were "*Not relevant at all*". In addition, we report median relevancy scores of all the generated suggestions. We also observe the effect of different factors, for example, across four different categories of the data inputs as discussed above (i.e., in terms of the dataset we used, and LM utilized in iZen) on the relevancy scores using statistical methods.

[5] CUREB-B Ethics Clearance ID: 118094

5.6.1.4 Automatic Metrics

To the best of our knowledge, there are no standard automatic metrics that can be leveraged to evaluate the "relevancy" of the generated suggestions against the original input statement as required by the proposed task ECSGen. Furthermore, since this is a newly proposed task, there is no precedence exists in conducting such evaluations. However, based on our experience in researching this area and developing ECSGen along with iZen, and some related literature from other domains (e.g., [255]), we selected a few metrics with available technical solutions to calculate them automatically. These metrics do not necessarily relate to the "relevancy" of the suggestions; instead, we expect them to represent the general quality of the generated suggestions. These include readability, grammaticality, and positivity. We consider these automatic metrics as complementary to the human evaluation discussed above.

5.6.2 Tools

5.6.2.1 Software tools for iZen:

For iZen we utilized several software tools and machine learning models. These included:

- Python and standard libraries [265]
- Language Tool Python [266]
- Profanity Filter [260]
- HuggingFace library [267]
- Grammar correction model: T5 Grammar Correction model [248]
- Toxicity model: Detoxify [256]
- Cardiff NLP's "Twitter-roBERTa-base" pretrained model for Emotion Recognition [220], [257]
- "DistilBERT-base-uncased-finetuned-SST-2" pretrained model for Sentiment Analysis [258]
- Large generative language models: GPT2 [128], OPT [130]

5.6.2.2 Survey Tools:

We used Qualtrics [261] to design and host our survey. The survey was distributed using Amazon Mechanical Turk (MTurk) [262].

We utilized several tools to analyze the survey results. These included:

- IBM SPSS [268]
- Microsoft Excel [269]
- Python and standard libraries [265]
- Pandas [270]

5.6.2.3 Automatic Metrics Tools:

For readability scores, we used a Python package named TextDescriptive [271]. For grammar check, we used another Python package name Language Tool Python [266].

For Sentiment Analysis, we used Cardiff NLP's Twitter-roBERTa-base for the Sentiment Analysis model [257] along with the HuggingFace library [267]. We also used Python and Excel for analyzing and visualizing some of these results.

5.6.2.4 Hardware Specification:

We used cloud hardware resources for our machine learning related workloads. These cloud hardware resources were available as part of our Google Colab Pro subscription [250]. This subscription provided us with the following hardware specification (maximum available):

- Processor: Intel Xeon CPU, 2.20 GHz (12 Cores)
- RAM: 87 GB
- GPU: NVIDIA A100 SXM4 40GB

For other compute workloads, we used a Macbook Pro (2019, 16-inch) with the following configuration:

- Processor: Intel Core i7, 2.6 GHz (6 Cores)
- RAM: 16 GB
- GPUs:
    - AMD Radeon Pro 5300M 4 GB
    - Intel UHD Graphics 630 1536 MB

5.6.3 Datasets

Since this study introduces a new NLP task - ECSGen, we created new datasets as part of this study to perform the task. We created two datasets: (1) Dataset generated using OPT (we name it: ECSGen-Data-OPT), and (2) Dataset retrieved from Twitter (we name it ECSGen-Data-TWT). We describe these datasets below:

5.6.3.1 ECSGen-Data-OPT

ECSGen-Data-OPT dataset is generated using OPT [130]. We chose OPT as it is one of the most advanced and latest general-purpose LMs available publicly. Specifically, we leveraged the "opt-1.3b" [249] variant due to its compatibility with our available hardware resources. As prompts for OPT, we used the specific "beginning part" texts mentioned in section 5.5.1 in order to generate input sequences with the desired format mentioned in the same section. Using random seeds, we generated multiple outputs for the same "beginning part" text. After the generation, we inspected every output in order to make sure the data is clean (e.g., does not include vulgarity, or obscenity).

This dataset contains 200 texts that begin with one of the "beginning part" texts mentioned in section 5.5.1. Each text follows the following pattern (as described in section 5.5.1): "*I am <express emotion> <causal conjunction> <cause>*". The static prompts given to OPT already included the initial parts: "*I am <express emotion> <causal conjunction>*" and OPT was tasked with generating the "*<cause>*" part. OPT was configured in a manner to make sure generated texts are simple, one sentence long etc. (similar criteria used for the "Zen" component of "iZen" as described in section 5.5.2.2. In summary, each datapoint in this dataset has "I" as the main subject, and the main subject expresses a negative emotion (see section 5.5.1 for more information) along with a cause associated with the expressed emotion. To benefit researchers with similar interests, we make this dataset publicly available[6] along with the top 3 generated suggestions and their relevancy scores based on the responses from our human evaluators.

5.6.3.2 ECSGen-Data-TWT

ECSGen-Data-TWT dataset is developed using data retrieved from Twitter [272] using existing libraries [273]. This dataset has similar qualities as ECSGen-Data-OPT. For instance, each text in

[6] https://github.com/riyadhctg/ECSGen-and-iZen

this dataset also begins with one of the "beginning part" texts mentioned in section 5.5.1. This was achieved by using the "beginning part" texts as query strings.

Similar to ECSGen-Data-OPT, we performed data preprocessing in order to ensure this dataset adheres to the data quality and format required to perform the proposed task with the iZen. For example, final datapoints in ECSGen-Data-TWT are one sentence long and grammatically correct, they start with "*I*", and each of them expresses a negative emotion with an associated cause. In summary, ECSGen-Data-TWT is similar to ECSGen-Data-OPT in terms of the content and quantity (i.e., 200 total datapoints); the only distinction being the source of the content. For ECSGen-Data-OPT, the content source is OPT, and for ECSGen-Data-TWT, the content source is Twitter.

5.7 Results

5.7.1 Automatic Metrics Results:

As discussed in the prior sections, we leveraged some automatic metrics in order to understand the general linguistic quality of iZen's generated suggestions. We discuss them below.

We found that about 80% of the generated suggestions had no grammatical errors. Approximately 15% contained 1 grammatical error and another 3% contained 2 grammatical errors. Only the remaining 2% had 3 or more grammatical errors.

We have used several standard metrics to analyze the readability of the generated suggestions (e.g., [217], [218]). Most of these metrics evaluate the readability of a text by age of the reader. Typically, a text is deemed easily readable by these metrics when it is easy to read for 13-15 years old who are proficient in English [274]. We use this threshold in reporting our findings on the readability of our generated suggestions. For the Flesch Reading Ease metric [217], this threshold means a score of 60 or above. Approximately 97% of the generated suggestions scored 60 or above in this metric. For Flesch Kincaid Grade [274], a score of 8 or below means it is easily readable. About 92% of generated suggestions had a score of 8 or below. For Gunning Fog metric [274], a good readable score is 13 or below, and approximately 94% of our generated suggestions had a score of 13 or below in this metric. Similarly, according to Automated Readability Index (threshold 9 or below), Coleman Liau Index (threshold 8 or below), and Lix metric (threshold 40 or below)

[275], about (in the same order) 91%, 93%, and 92% of iZen's generated suggestions were easily readable. Table 17 summarizes these scores.

Readability Metrics	Easily readable (% of all generated suggestions)
Flesch Reading Ease	97%
Flesch Kincaid Grade	92%
Gunning Fog	94%
Automated Readability Index	91%
Coleman Liau Index	93%
Lix	92%

Table 17: Automatic metrics score of the generated suggestions

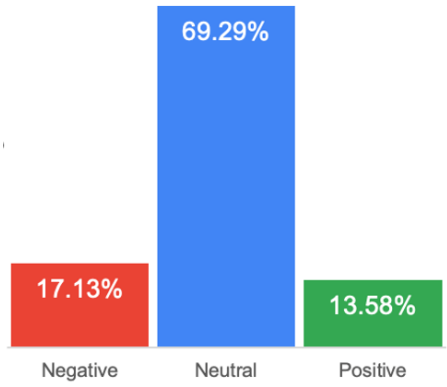

Figure 30: Sentiment of the generated suggestions

In addition to basic linguistic quality, we also evaluated the generated suggestion on their sentiment. Since iZen generates suggestions to address a negative emotion expressed in a simple statement, we expect that the majority of the generated suggestions will be either neutral or positive instead of having negative sentiment. As shown in Figure 30, we found that more than 82% of the generated suggestions were either neutral (about 69%) or positive (about 14%).

5.7.2 Human Evaluation Results:

5.7.2.1 Relevancy scores of the generated suggestions

As discussed earlier, our study involved 800 input statements, and for each of them, iZen generated three suggestions, summing up to a total of 2400 suggestions. Our survey participants evaluated these suggestions using a 5-point Likert scale (numeric labels were 1-5, mentioned in the bracket beside each text label): "*Not at all relevant*" (1), "*Slightly relevant*" (2), "*Somewhat relevant*" (3), "*Moderately relevant*" (4) and "*Very relevant*" (5).

As shown in Figure 31, among 2400 suggestions, our participants found 17.5% of them "*Very relevant*", 17.8% "*Moderately relevant*", 19.4% "*Somewhat relevant*", 18.5% "*Slightly relevant*" and 26.8% "*Not relevant at all*". In order words, about 73% of the generated suggestions were slightly or more relevant; approximately 55% were somewhat or more relevant; about 35% were moderately or more relevant.

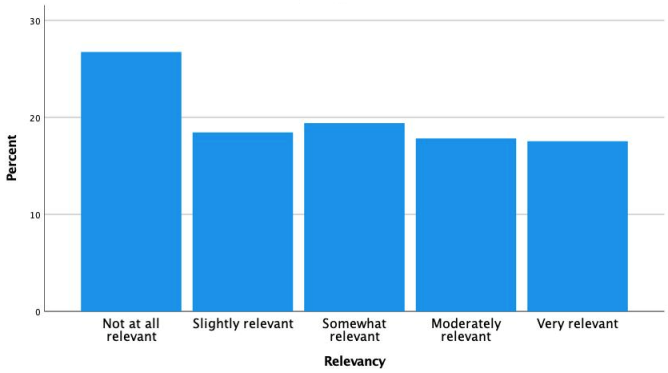

Figure 31: Overall relevancy of all the generated suggestions.

For each of the 800 input statements, iZen generated three suggestions. We wanted to understand how many of the 800 input statements had at least 1 or more generated suggestions that achieve a minimum threshold in terms of the relevancy score. We found that for 800 input statements, 91.25% had at least 1 slightly or more relevant suggestion, 80% at least 1 somewhat or more relevant suggestion, 68.25% at least 1 moderately or more relevant suggestion, and 38% at least 1 very relevant suggestion. We also observed that 76.75% of the input statements had at least 2 slightly or more relevant suggestions, 59.25% at least 2 somewhat or more relevant suggestions, 28% at least 2 moderately or more relevant suggestions, and 10.875% at least 2 very relevant suggestions. We found that all three suggestions for each input statement were slightly or more relevant for 51.75% of the input statements, somewhat or more relevant for 25.125% of the input statements, moderately or more relevant for 9.875%, and very relevant for 3.75% of the input statements. Lastly, we found that for 8.75% of the input statements, all three suggestions generated by iZen were irrelevant. We summarize and illustrate these findings in Table 18 and Figure 32.

Relevancy	Percentage
At least 1 slightly or more relevant	91.25
At least 1 somewhat or more relevant	80
At least 1 moderately or more relevant	68.25
At least 1 very relevant	38

At least 2 slightly or more relevant	76.75
At least 2 somewhat or more relevant	59.25
At least 2 moderately or more relevant	28
At least 2 very relevant	10.875
All slightly or more relevant	51.75
All somewhat or more relevant	25.125
All moderately or more relevant	9.875
All very relevant	3.75
None is relevant	8.75

Table 18: Relevancy of the generated suggestion per input statements

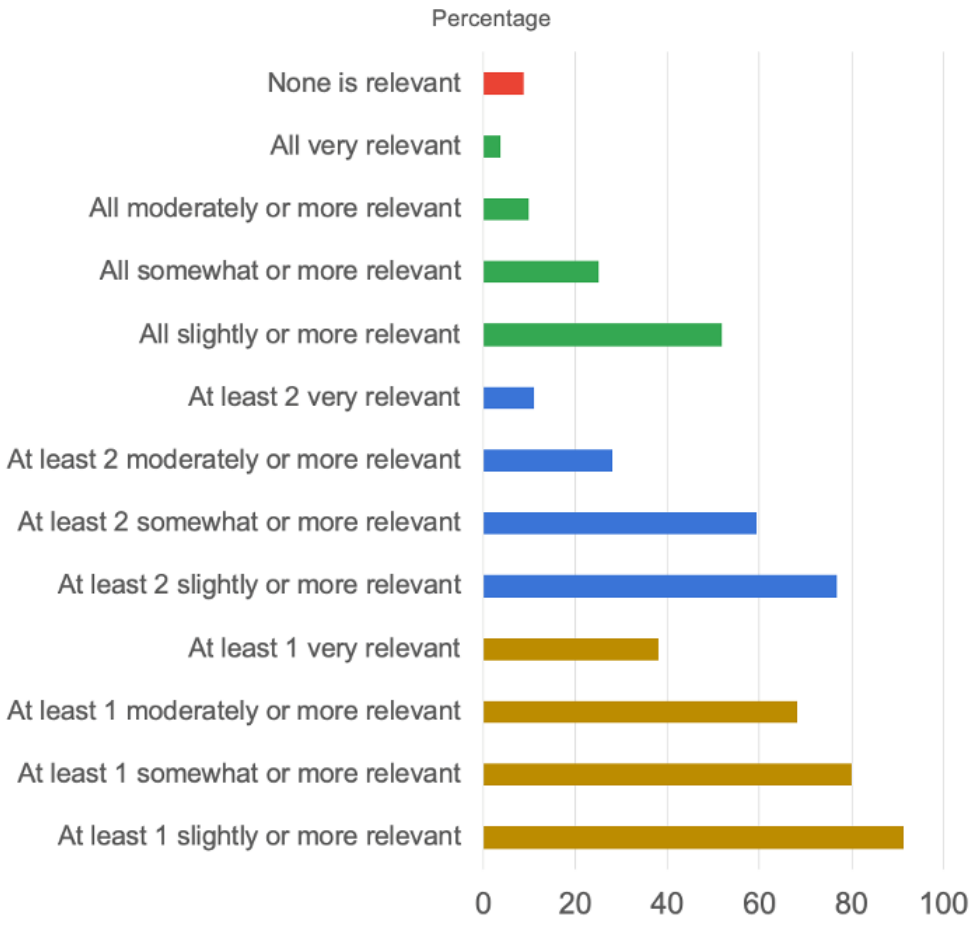

Figure 32: Relevancy of the generated suggestion per input statements.

5.7.2.2 Comparison of relevancy scores across different factors

5.7.2.2.1 Contextual richness of the input statement

For each input statement, along with evaluating suggestions using the Likert scale, we also asked our participants to evaluate whether or not the original input statement contained enough context

or information in order to provide a relevant suggestion. Participants answered this question using "*Yes*" (contains enough context), "*No*" (does not contain enough context), or "*Maybe*" (unsure whether or not the statement contains enough information). Among 800 input statements (used to generate 2400 suggestions), 446 received a "*Yes*" response, 139 statements a "*No*" response, and 215 received a "*Maybe*" response from our participants. We performed further analysis to understand how this score relates to the relevancy score for the generated suggestions. We observed that overall when the input statement contained enough context, the generated suggestions were likely to be more relevant. A p-value of less than 0.001 in a Chi-Square test [77] confirms this impact of context contained within the input statement on the generated suggestion. We illustrate these results in Figure 33 and Figure 34.

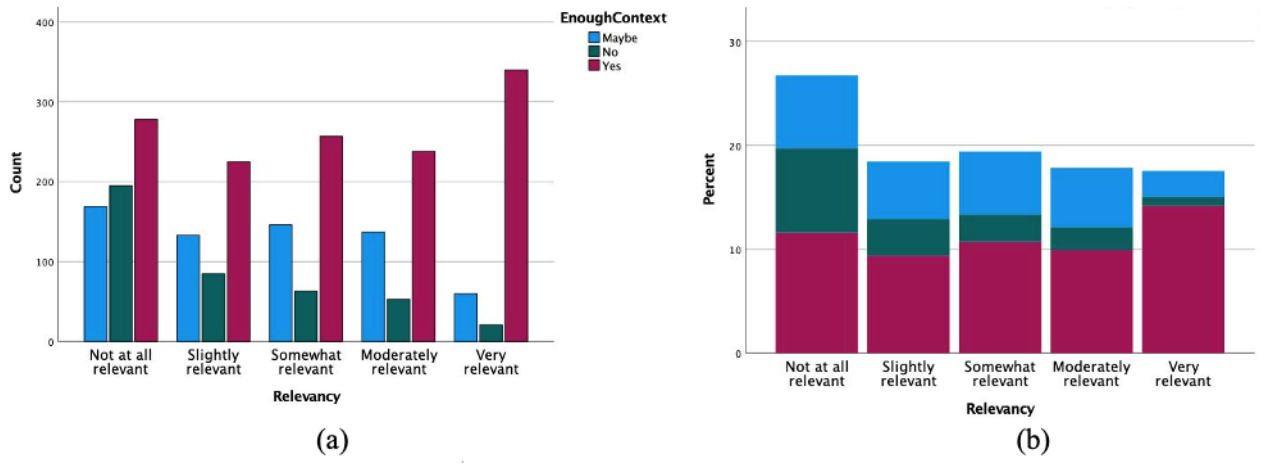

Figure 33: Variation of the relevancy of the generated suggestion based on the contextual richness of the input statement. (a) Grouped bar chart (where the y-axis represents the "count" of the suggestions), (b) Stacked bar chart (where the y-axis represents the "percentage" of the suggestions)

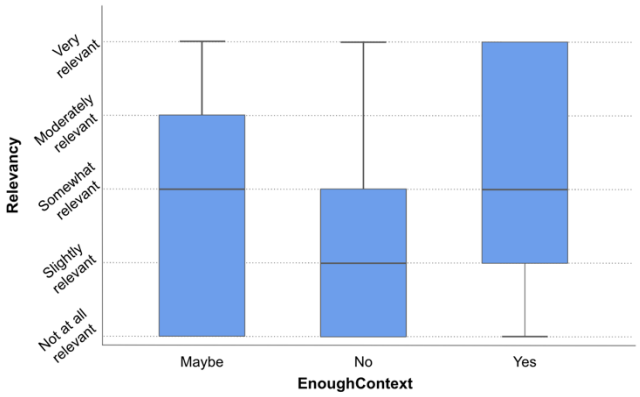

Figure 34: Boxplot to show the variation of the relevancy of the generated suggestion based on the contextual richness of the input statement.

5.7.2.2.2 Input Source: OPT vs. Twitter

We used the language model (OPT) [272] and Twitter [272] as the source for our input statements. We performed a Chi-Square test to understand if there is any significant impact of the input source on the relevancy of the generated suggestions. A Chi-Square test, achieving a p-value of less than 0.05 revealed that the input source had a significant impact on the relevancy score achieved by the generated suggestions. Overall, when the data source for the input statements was Twitter, participants found iZen's generated suggestions to be less relevant compared to when the data source of the input statements was OPT language model (LM). We visualize these results in Figure 35 and Figure 36.

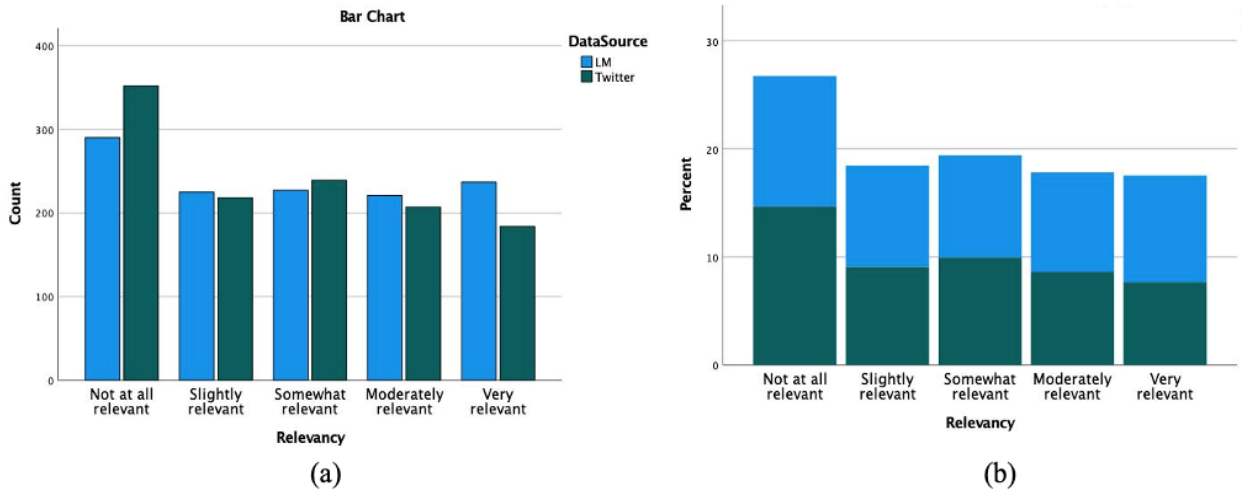

Figure 35: Variation of the relevancy of the generated suggestion based on the data source of the input statement. (a) Grouped bar chart (where the y-axis represents the "count" of the suggestions), (b) Stacked bar chart (where the y-axis represents the "percentage" of the suggestions)

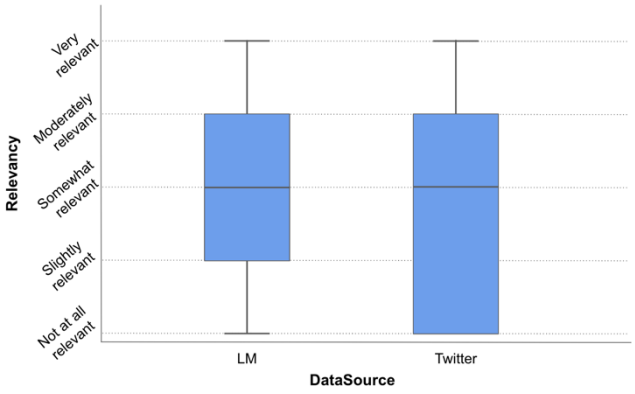

Figure 36: Boxplot to show the variation of the relevancy of the generated suggestion based on the data source of the input statement.

5.7.2.2.3 LM used in Zen: OPT vs. GPT2

iZen leveraged either GPT2 or OPT language model (LM) for its Zen component. We perform a Chi-Square test to understand if there was any significant impact of the choice of LM for Zen on the relevancy score of the generated suggestions. The Chi-Square test result (p-value of 0.478) suggests that there was no significant impact of this choice on the Likert score of the generated suggestions. We illustrate these findings in Figure 37 and Figure 38.

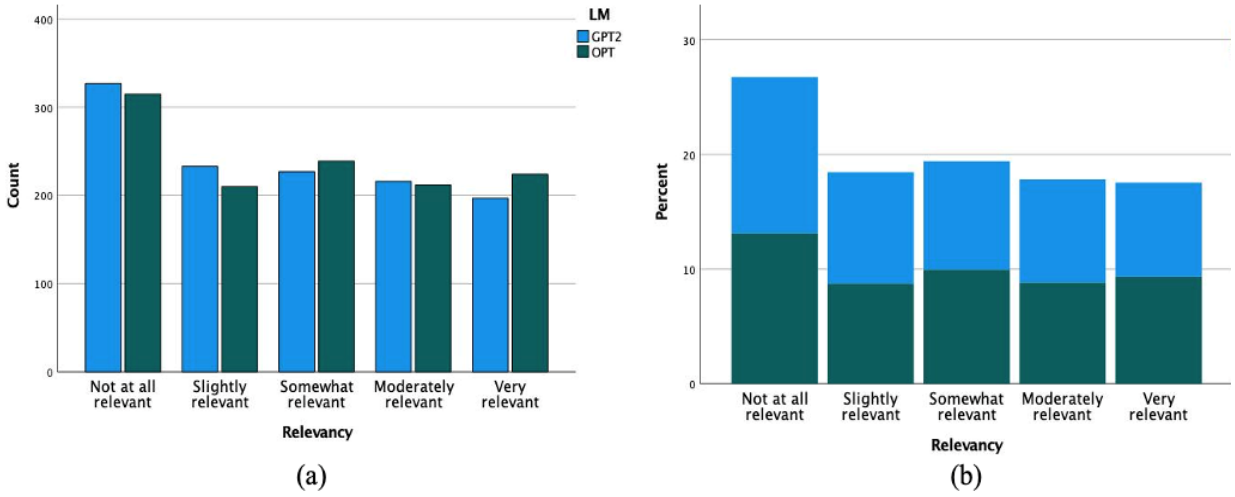

Figure 37: Variation of the relevancy of the generated suggestion based on the LM used in Zen. (a) Grouped bar chart (where the y-axis represents the "count" of the suggestions), (b) Stacked bar chart (where the y-axis represents the "percentage" of the suggestions)

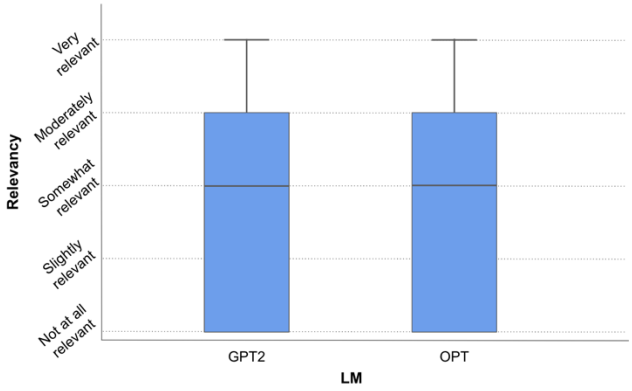

Figure 38: Boxplot to show the variation of the relevancy of the generated suggestion based on the LM used in Zen.

5.8 Discussion

5.8.1 Generated suggestions are easily readable and contain minimal grammatical errors

Based on the automatic metrics, namely the count of grammatical errors and various readability metrics, we observed that iZen's generated suggestions had sound linguistic quality. About 95% of the generated suggestions had 1 (15%) or 0 grammatical (80%) errors which demonstrate the

grammatical correctness of iZen's generated suggestions. More than 90% of iZen's generated suggestions were deemed easily readable by all the readability metrics used in this study.

5.8.2 Generated suggestions contain minimal negative sentiment

We observed that the generated suggestions contained minimal negativity. Most of iZen's generated suggestions were neutral (69.29%), while 13.58% were positive and 17.13% were negative. While we expect that most of the generated suggestions to not have negative sentiment, we also believe that a negative sentiment (as determined by the Sentiment Analysis technique used in the study) may not necessarily mean a bad suggestion and vice versa. This is because a typical Sentiment Analysis technique may consider something negative based on the constituent words, while the suggestion may actually be meaningful in the context of the ECSGen task. For example, in our analysis, for the statement, "*I feel angry because of the information*", one of iZen's generated suggestions was, "*Do not click on those links*". While the sentiment of this suggestion was determined to be negative, the suggestion seems meaningful based on the original statement. While non-negativity of the suggestions may be an important factor, based on our observation, we opine that it is not a primary indicator of a suggestion's relevancy.

5.8.3 Generated suggestions are mostly relevant with varying degrees of relevancy

Results from the human evaluation of iZen's generated suggestions suggest that 26.8% of the generated suggestions were completely irrelevant, indicating that the remaining 73.2% of the suggestions were relevant to the emotion expressed in the input statement. The relevancy of these 73.2% generated suggestions was of varying degrees (17.5% very relevant, 17.8% moderately relevant, 19.4% somewhat relevant, 18.5% slightly relevant). The median relevancy score for all generated suggestions was "*Somewhat relevant*".

We also observed that for 91.25% of input statements, iZen generated at least 1 slightly or more relevant suggestion. For 76.75% of input statements, iZen generated at least 2 slightly or more relevant suggestions. At least 1 suggestion was somewhat or more relevant for 80% of the input statements, and at least 2 suggestions had a similar level of relevancy for 59.25% of the input statements. 68.25% of input statements yielded at least 1 moderately or more relevant suggestion, and 28% of the input statements had at least 2 moderately or more relevant suggestions. For

51.75% of the input statements, all three generated suggestions were slightly or more relevant. Only 8.75% of the input statements received no relevant suggestions at all.

In summary, we believe that the following achievements by iZen's generated suggestions indicate that iZen demonstrated promising results in performing the ECSGen task:

- A median relevancy score of "*Somewhat relevant*",
- 73.2% of all the generated suggestions are slightly or more relevant,
- 91.25% of the input statements received at least 1 slightly or more relevant suggestions,
- 80% of the input statements received at least 1 somewhat or more relevant suggestions.

5.8.4 Contextually rich input statements resulted in more relevant suggestions

Our results suggest that iZen can generate more relevant suggestions when the input statement has rich contextual information. We see in Section 5.7.2.2.1 that less contextually rich input statements generated most of the "*Not at all relevant*" suggestions, while most of the "*Very relevant*" suggestions were generated from input statements with rich contextual information. As supported by the statistical test, we conclude that the richness of contextual information in the input statement is an important factor for iZen to generate relevant suggestions.

5.8.5 OPT and Twitter as data sources for input statements impacted the relevancy of the generated suggestions

We observe that the two data sources (OPT [130], Twitter [272]) used in this study for input statements impacted the relevancy score of the suggestions generated from them. In our experimental settings, iZen generated more relevant suggestions for input statements generated from OPT than compared to input statements extracted from Twitter. We believe that this outcome warrants further investigation in order to identify what exact factor(s) contributed to the difference in relevancy score for input statements from these two sources.

5.8.6 GPT2 and OPT as iZen's core generative LM performed on par in terms of relevancy score

Our results suggest that both OPT [130] and GPT2 [128] performed similarly to iZen's core generative language model. While OPT is more recent and arguably more advanced compared to

GPT2, for the ECSGen task, their performance within iZen's architecture was found to be comparable. We believe that these findings warrant experimenting with more language models within the Zen component in order to observe how they impact iZen's generated suggestions' relevancy.

5.9 Risks, Limitations and Future Work

5.9.1 Potential risks posed by iZen

In the last safeguard component, Guardrail, we only used the "profanity filter". It can be argued that this single filter may not be sufficient as the last guardrail. There can be suggestions generated by LLMs that may not have any profanity in them but still may be inappropriate. For example, LLMs can generate insensitive suggestions that may potentially contradict the objectives of the ECSGen task that aims to generate suggestions that "mitigates" negative emotions and not aggravates them. This warrants further investigation in future and can be an interesting research area to explore.

With all the filters in place, the possibility of iZen and similar systems generating unacceptable suggestions cannot be fully ruled out. As a standard practice, these types of systems, if released publicly, should come with substantial warnings about the potential dangers it can pose for their users. These types of systems often benefit from user feedback, from which they can continuously learn and become better at avoiding undesirable situations such as generating inappropriate suggestions. An intuitive way to report unacceptable issues may help these systems (when publicly released) learn better and faster.

Another measure that can be taken to improve the resilience of iZen from generating inappropriate suggestions is input moderations. In iZen we included Guardrail which is a safeguard component that comes at the end of the pipeline. In addition to this, we can include another safeguard component at the beginning of the pipeline to moderate the inputs to iZen. Input moderation can have multiple benefits since it can be a decoupled and a more lightweight component which can restrict entry to the main iZen pipeline, thereby avoiding potentially unnecessary suggestion generation and detecting inappositeness early.

The above points only serve as some examples of the potential risks posed by iZen and it is not an exhaustive list. When developing such AI systems for public release, we must diligently consider the potential risks they pose and take necessary steps to mitigate them.

5.9.2 Input constraints

iZen's development was initiated with defining several input constraints as mentioned in the paper. As a result, iZen can only perform the ECSGen task under certain conditions. This restricts iZen's capability. While more rules can be included in iZen's PromptGenie component to reduce the constraints, it may not be scalable. A major task of this component is to extract the cause of the emotion. A significant portion of the input constraints were added to enable the rule-based PromptGenie component to perform this extraction. One potential alternative to this can be leveraging the existing Emotion-Cause Extraction (ECE) techniques within the ECA domain. Introduction of such components within iZen's PromptGenie component may allow significantly reducing the number of input constraints that currently exist. We believe there is ample opportunity for further research in this area in order to reduce the number of input constraints in iZen to enable it to support a wide range of input statements.

5.9.3 Relevancy score

In our experimental setting, iZen generated 3 suggestions for 1 input statement. Based on iZen's performance (e.g., at least 1 of the 3 suggestions was somewhat or more relevant for 80% of the input statements), we can state that this is a promising achievement by a framework that did not receive any specific training to perform this task. However, we believe this performance can be further enhanced by devising a more innovative and sophisticated framework. In addition to designing completely new frameworks to perform ECSGen, one can explore customizing iZen due to its conceptual simplicity and modularity. For example, the Zen component can effortlessly leverage other similar language models to generate candidate suggestions, which is a worthwhile investigation. The same is true for the Eliminator and Guardrail components where the models and libraries used can be easily replaced and extended.

5.9.4 Leaner system

iZen's superiority comes from the fact that it can perform the ECSGen task without receiving specific training to perform this task. However, this comes with a few drawbacks. For example,

iZen includes a reasonably complicated post-generation filtering mechanism that is comprised of multiple pre-trained machine learning models. This increases the size and complexity of the architecture. There is ample opportunity to innovate in this area in order to devise a leaner and lightweight solution that can perform ECSGen with similar or better performance.

5.9.5 Alternative design for iZen

Our proposed design for iZen in this chapter assumes that 1 or more suggestions out of the 11 generated by the Zen component for each input statement are plausible that meet the desired criteria. While this is not an unrealistic assumption for the advanced large LMs given an appropriate prompt exists, particularly with a stochastic decoding strategy that is used in this study [237], and that we have observed encouraging performance by iZen, however, it is still constrained by a fixed number (i.e., 11) of candidate suggestions generated by Zen. We believe this offers an opportunity to explore alternative designs for iZen, particularly the one where Zen can be invoked for iterative suggestion generation until a (or more) suitable output is found. Needless to say, such a design still requires a threshold that offers an exit path in case where no suitable output is found in order to avoid an infinite loop. This alternative design also provides an opportunity to include Guardrail with Eliminator to streamline the framework.

Another potential modification to iZen's current design can be inside the Eliminator component. Eliminator's Selective Deduction Unit, E_{SD} follows a greedy algorithm approach which was designed to address the critical issues first (i.e., toxicity, grammatical errors). There is opportunity to explore alternative algorithms for this selection process. For instance, sum all the scores from the Eliminator subcomponents can be considered together using appropriate weights in order to select the top three suggestions. We believe a worthwhile future work would be to explore such alternative designs for iZen.

5.9.6 Relevant automatic metrics

For ECSGen, we primarily relied on human evaluation to understand the relevance of the generated suggestions. Human evaluation introduces several complicated and time-consuming steps within the entire research project. In our observation, this is not only an issue for ECSGen but a common issue for generative tasks in general. An aspirational future work can be developing relevant automatic metrics for the ECSGen task to measure the relevancy of the generated suggestions.

5.10 Conclusion

In this study, we propose ECSGen, a new NLP task within the ECA domain. The purpose of this task is to generate relevant suggestions to mitigate the cause of a negative emotion expressed in a given input statement. We also devise a technical framework named iZen to perform this task in a zero-shot manner within certain constraints. We curate two new datasets to conduct our experiment with iZen. iZen generates three suggestions for each input datapoint. We perform human evaluations to understand the relevancy of these generated suggestions. The median relevancy score of the suggestions generated by iZen was "*Somewhat relevant*". More than 90% of the input statements in our experiments received at least 1 slightly or more relevant suggestion, and about 80% received at least 1 somewhat and more relevant suggestion. More than 70% of all the generated suggestions were slightly or more relevant. These are promising results suggesting iZen's decent capability to perform the proposed ECSGen task. However, we acknowledge that there is room for improvement for iZen on several fronts. These include but are not limited to, generating suggestions that are more relevant, reducing iZen's input constraints so that a large variety of input statements can be supported, and making iZen more lightweight, especially in terms of the number of pretrained models required to perform the task. We plan to tackle these challenges in our future studies.

Chapter 6: Towards Automatic Evaluation of NLG Tasks using Conversational Large Language Models

In the previous two chapters, we introduced two new open-ended Natural Language Generation (NLG) tasks in the domain of Emotion-Cause Analysis (ECA). A common challenge we encountered in both studies was the automatic evaluation of the techniques that performed these tasks. Due to the open-ended nature of these tasks, these are difficult to assess in a rule-based manner, often requiring tedious evaluation by human participants. Consequently, for the first NLG task we proposed, named Emotion-Cause Generation (ECG), due to the limited scope of the study, we primarily restricted ourselves to the available automatic metrics, all of which only supported assessing the general linguistic quality of text. For the second NLG task we proposed, named, Emotion Cause mitigating Suggestion Generation (ECSGen), we relied on human participants as our primary mode of evaluation. Conducting such tedious evaluations by human participants often reduces the pace of research, which is particularly apparent for a fast-advancing research fields like NLG. We observe that this is a challenge experienced by many researchers working on NLG problems. In fact, evaluating the quality of machine generated open-ended texts is a long-standing challenge in Natural Language Processing (NLP). Although the NLG field has seen rapid advancements in recent years, a promising and widely adopted automatic evaluation technique for NLG tasks is yet to be developed. In this chapter, we propose leveraging Conversational Large Language Models (LLMs) as automatic evaluators for open-ended NLG tasks. To demonstrate the viability of our proposal, we perform several experiments with three common NLG tasks as well as the two NLG tasks that we introduced in this thesis.

The work presented in this chapter has been accepted into one conference[7].

6.1 Introduction

With the advent of Transformers [95] and large pretrained language models (LLM) (e.g., [129]) in recent years, there has been a plethora of research work in Natural Language Processing (NLP)

[7] [Accepted] Riyadh, M., & Shafiq, M. O. (2023). Towards Automatic Evaluation of NLG Tasks using Conversational Large Language Models. In 19th IFIP International Conference on Artificial Intelligence Applications and Innovations.

that contributed significantly to the advancement of this field, particularly in the area of Natural Language Generation (NLG) [277]. While there have been rapid advancements in the NLG field, the difficulty in evaluating them has become more prominent. Unlike many NLP tasks such as classification tasks like Sentiment Analysis, NLG tasks cannot be easily evaluated with automatic evaluation techniques. This is due to the nature of the NLG tasks, many of which can be open ended. Some researchers have attempted to develop automatic evaluation techniques for NLG tasks [216], [278]. However, none of them have received widespread adoption primarily due to their poor generalizability to evaluate open ended and diverse NLG tasks. As a result, a vast majority of the NLG tasks still require performing tedious and lengthy evaluations by human participants.

A recent addition to the LLM technology is ChatGPT [131], which is closely related to the InstructGPT [279] model and fine-tuned from a GPT-3.5 variant [280]. As the name suggests, ChatGPT is a conversational LLM that can provide a detailed answer to a variety of questions. It is trained to understand many simple and complex instructions and execute those instructions effectively. Guo et al. [281] experimented with ChatGPT's capability to perform a variety of NLP tasks. In this study, we tap into the similar capabilities of ChatGPT but with the objective of understanding its ability to be an automatic evaluator for NLG tasks. In addition to some common NLG tasks, we focus on the domain of Emotion-Cause Analysis (ECS) [232], which consists of several tasks related to the cause of an expressed emotion in a given text. Our experiments cover all the NLG tasks within the ECA domain that are currently available: Emotion-Cause Generation (ECG) [see Chapter 4] and Emotion-Cause mitigating Suggestion Generation (ECSGen) [see Chapter 5]. Through a series of experiments, we attempt to understand if ChatGPT can be an effective automatic evaluator for various NLG tasks, particularly the ones that do not require specific domain knowledge and that typically need to be evaluated by human participants.

6.2 Contributions

The main contributions of this study include:

- We propose leveraging Conversational LLMs as automatic evaluators for the NLG tasks, especially for open-ended tasks that do not require domain specific knowledge.

- We perform a series of experiments to understand ChatGPT's (a recently released advanced conversational LLM) ability as an automatic evaluator for three common NLG tasks: Paraphrasing, Question-Answer, Story-cloze test (i.e., evaluates models' common-sense reasoning capability).
- We also experiment with two ECA related NLG tasks: ECG, and ECSGen. These are currently the only two NLG tasks available within the domain of ECA. Consequently, our experiments on the usage of conversational LLM as an automatic evaluator cover all ECA related NLG tasks that exist today.

6.3 Related Studies

The evaluation of NLG systems has conventionally been conducted by human evaluators [282]. In such procedures, human evaluators are presented with generated texts from the NLG systems and human-written texts. The performance of the NLG system is determined by comparing the ratings assigned to each. This evaluation approach, which was first introduced in the mid-90s [283], continues to be utilized in the present day in various forms for the majority of the NLG tasks. Researchers later suggested that evaluating NLG systems by comparing the generated texts to a corpus of human-written texts offers the potential for faster, more cost-effective and scalable evaluation [284]–[286]. Despite its advantages, this method has faced criticism, with objections such as that the generated texts may differ from corpus texts while still meeting the criteria of successfully completing an NLG task [287].

Some other researchers developed a group of alternative automatic metrics called word-based metrics (WBM). WBMs typically evaluate the similarity between the output text produced by the systems and human-generated reference texts based on various criteria [288]. The closer the similarity between the output and reference texts, the higher the score obtained from the metric. BLEU [213], ROUGE [215] etc. are some common examples of such metrics. However, none of these metrics can perform the tasks typically done by human evaluators due to their limited scope.

In addition to WBM, automatic metrics related to NLG also include readability metrics such as the Flesch Reading Ease score [217] that quantifies the "reading ease" of a text based on the number of characters per sentence, words per sentence, and syllables per word. Grammatical correctness is also another automatic metric that some NLG studies tend to use to understand if an output by

a machine learning model is grammatically correct [222]. More recently, there has been a growing interest in the use of Transformer-based metrics for evaluating NLG tasks. They can be broadly divided into two categories: reference-based models that require human references (e.g., [289], [290]) and reference-free models that do not (e.g., [291], [292]). The latter category is more cost-effective to implement. However, some researchers pointed out that these metrics are still inadequate for evaluating specific NLG tasks without infusing task specific knowledge into them – which can be burdensome and increase the scope and complexity of any given study [293]. This could be one of the reasons why older and less accurate metrics such as BLEU continue to dominate the field of automatic evaluation of NLG tasks despite their limitations [294], [295].

Due to all these limitations of the existing automatic evaluation techniques for NLG tasks, particularly for tasks that are more open ended in nature such as paraphrasing, most researchers still opt for human evaluation as the only available but tedious and time-consuming alternative.

6.4 Conversational LLMs as Automatic Evaluators for NLG Tasks

We propose leveraging conversational LLM as an alternative automatic evaluator for many NLG tasks, especially the ones that do not require specific domain knowledge, and that typically requires human evaluators. Evaluating NLG tasks with human evaluators require the evaluators to understand the tasks with simple instructions and execute those tasks accordingly. For example, to evaluate the meaningfulness of a summary of a given news article, human evaluators are typically given instructions on how to evaluate the relevance of such summaries in relation to the original news article. Based on that, they may be tasked with ranking the relevancy of those summaries. We propose that we can utilize conversational LLM to conduct such evaluations. Particularly, in this study, we focus on ChatGPT [131], a recently released conversational LLM to understand its ability as an automatic evaluator for open-ended NLG tasks.

As we demonstrate in the upcoming sections, ChatGPT is similarly capable of understanding the instructions typically given to human participants and then completing the given task. In the context of this study, "ability of understanding the instructions" refers to the fact that ChatGPT is able to follow the textual instructions given to it in order to carry out the tasks we evaluated. For example, we can input the following textual instructions (i.e., similar to natural language humans use to communicate with each other) into ChatGPT: "*paraphrase this sentence: <I could not go*

to the office today because of sudden illness>", then, it can follow these instructions and carry out the anticipated task. This capability, in our opinion, makes conversational LLMs like ChatGPT suitable candidates to become automatic evaluators for many NLG tasks.

To use ChatGPT for such evaluations, the instruction set of each task needs to be designed carefully, ideally through experimenting with ChatGPT and by observing its response. These instruction sets are often called "prompt" in the context of LLMs [243]. After an effective prompt template is determined for a given NLG task, prompts for each datapoint can be generated automatically by developing simple custom logic in programming languages such as Python [265]. These prompts then can be provided to ChatGPT using the available interface and then its response can be collected subsequently. In the following section, we conduct a series of experiments to understand ChatGPT's capability as an automatic evaluator for some NLG tasks.

6.4.1 Evaluation

In this study, we evaluate ChatGPT as an automatic evaluator for the following five NLG tasks: (a) Paraphrasing, (b) Question-Answering, (c) Story-cloze Test [296], (d) ECSGen [see Chapter 5], (e) ECG [see Chapter 4].

6.4.1.1 Procedure

We perform five small scale experiments to evaluate ChatGPT for the five selected NLG tasks. Each experiment has some specificities, but they all share some common procedures, which we discuss below:

- We take a small sample of data (i.e., about 100 datapoints) from an existing dataset for each task. This is because we leveraged ChatGPT's Research Preview [131] for this study which (at the time of this study) is accessible through a browser interface requiring tedious manual input. For sampling, we leverage different random sampling techniques including balanced and stratified sampling for different tasks to present diverse data compositions to ChatGPT.
- To use the dataset for our intended evaluation, we preprocess each of them differently depending on their original format and evaluation process. We auto-generate ChatGPT prompts for each datapoint for all five tasks using Python [265]. We use different prompt formats for each task, which we designed by iteratively experimenting with ChatGPT for the

best outcome in terms of aiding ChatGPT to understand the task as well as receiving an answer from it in a format that is easy to process for analysis.

- We enter these prompts into ChatGPT's interface and manually collect its response for each of the datapoint. This interaction with ChatGPT Research Preview occurred between Jan 26, 2023, and Feb 16, 2023.
- After that we compare ChatGPT's responses with the ones included in the datasets. To observe how ChatGPT performs as an automatic evaluator for different NLG tasks that can be assessed differently, we leverage diverse techniques such as classification accuracy, multiple choice type questions (MCQ), correlating Likert responses etc. We use Python [265], IBM SPSS [268], and Google Sheet [297] for our analysis.

Experiments for each task included some distinct procedures, particularly as it relates to the dataset used for each, how the task is conducted as well as the analysis methods we used for them. We discuss them below.

For the paraphrasing task, we take a sample from the GLUE dataset's Quora Question Pairs [298]. The dataset contains two one-sentence long questions. The task is to determine whether the questions are paraphrases of each other. To understand if ChatGPT can automatically evaluate if a paraphrase is correct, we frame this as a binary classification problem for analysis where ChatGPT attempts to determine the correct and incorrect paraphrases. ChatGPT's performance on this would indicate how it can perform as an evaluator for the NLG task when machine learning models generate a paraphrase for a given text (in this case, short text).

For the question-answering task, we utilize the WikiQA dataset [299] where each question has several correct and incorrect answers. The goal is to determine which answers are correct and which ones are not. For ease of analysis, we randomly sample some questions with one incorrect answer and some with one correct answer. We then frame this task as a binary classification problem for our analysis where ChatGPT's task is to determine which answers are correct for a given question and which are not. We expect this to indicate how ChatGPT can perform as an evaluator to understand a model's performance in generating answers for a given question.

Similarly, we leverage an existing dataset for the Story-cloze test [296] which also evaluates the common-sense reasoning of machine learning models. Each datapoint in this dataset contains a short story consisting of five sentences. For each, the first four sentences are provided in the correct

order. For the fifth sentence, there are two options, only one of them being correct. The task is to determine which option is the appropriate choice for the fifth or last sentence of the story. ChatGPT's performance in determining the correct fifth sentence in this task would indicate how well it can evaluate the models that are tasked with completing unfinished stories using common-sense reasoning.

The ECSGen dataset [see Chapter 5] contains 5-point Likert responses [264] from human evaluators for each "emotion-cause mitigating suggestion", representing the degree of relevance of the suggestion in relation to the cause of the emotion. The five levels of the Likert scale are: 1: Not at all relevant, 2: Slightly relevant, 3: Somewhat relevant, 4: Moderately relevant, and 5: Very relevant. The task for ChatGPT is to rank the sampled suggestions similarly. The dataset contains 3 suggestions for each statement. We sampled 120 datapoints; consequently, our sample contains 360 suggestions. For analysis, we frame this as a multi-category classification problem. Since the human response for each suggestion can be extensively subjective for this type of question, particularly with 5 intensity levels to choose from, we compare ChatGPT's response with the ones from human evaluators using various techniques such as correlation testing, classification accuracy at 5-levels, and classifications accuracy at 3 and 2-levels by recoding the Likert labels accordingly. We believe ChatGPT's performance in this experiment can indicate how well it can rank the relevancy of a suggestion that aims to mitigate the cause of an emotion expressed in a short text.

For the ECG task [see Chapter 4], we preprocess a sample from an existing Emotion-stimuli dataset [9] for our study. We replace the "cause-span" for each datapoint with an underscore ("_"). We convert the dataset to a comma-separated values format where the separated cause-span, and the original texts (without the cause-span) are placed into separate columns. Then we create a duplicate column from the "cause-span" column and randomize the order of this new column. Thus, for each datapoint, we have a correct cause and an incorrect cause for the expressed emotion. We remove the datapoints where we found duplicates between these two cause columns and where both causes seemed equally appropriate. We frame it as an MCQ-style binary classification task for ChatGPT where it attempts to pick the correct cause for each statement with an emotional expression. We expect this to indicate ChatGPT's ability to evaluate the meaningfulness or relevance of a generated cause in relation to the emotion expressed in a text. Please refer to the following table for examples of ChatGPT prompts used in this study for each NLG task.

NLG Task	ChatGPT prompts used in this study
Story-cloze test	Consider the following sentences from a short story: Betsy was about to celebrate her sixteenth birthday. Her parents surprised her with a nice dinner out and a small present. Happily, her parents gave her a monogram necklace. On the way home, Betsy misplaced the necklace in the car. Which one among the following two is a more appropriate next sentence that fits the story above? 1. Betsy felt really bad for the rest of the night. 2. Betsy wore the monogrammed necklace when she turned 14. Answer using only numbers 1 or 2. Do not write any other additional notes or explain your answer
Question Answering	Consider the following question: - what are garnishments? Is the following a correct answer to the above question? - A garnishment is a means of collecting a monetary judgment against a defendant by ordering a third party (the garnishee) to pay money, otherwise owed to the defendant, directly to the plaintiff. Answer with Yes or No. Do not write any additional notes.
Paraphrasing	Consider this statement: How do I start up a new cafe? Is the following a correct paraphrasing of the above statement? - What are requirements I would need to start my own cafe? Answer with Yes or No. Do not write any additional notes.
ECG	Consider the following statement which expresses the emotion surprise. The cause of this emotion is missing which is denoted by '_': - Martha replied and left her twin sister blinking in astonishment _ but which was now enticingly attractive Which of the following causes is contextually more meaningful for the emotion expressed in the above statement? Please consider the context of the statement above as the cause should not only be relevant to the emotion expressed, but also the context of the emotion: A. at a possibility that had not occurred to her before. B. at the lack of progress Corbett was making. If you are not sure, then just pick your best guess. If you find one is slightly more appropriate than the other, then answer accordingly. Do not write any additional notes. Answer with letter: A or B. No need to explain your answer or rewrite the cause.
ECSGen	Consider this statement: I feel bored because I have been here for so long. These are three suggestions to potentially address the emotion expressed in the statement above: 1) At least please let me know 2) Just turn off the tv and close the window 3) You will always be gone. Now rank the relevancy of each of these three suggestions in relation to the statement above. Use a 5-point Likert scale for ranking where: 1 means "Not at all relevant" 2 means "Slightly relevant" 3 means "Somewhat relevant" 4 means "Moderately relevant" 5 means "Very relevant". Put your ranking in an array in the same order as the suggestion above. Don't write any additional notes.

Table 19: ChatGPT Prompts used in this study for different NLG task

In this study, we evaluate ChatGPT, a recently released conversational LLM, as an automatic evaluator for a set of NLG tasks and report our findings. We do not compare ChatGPT's performance for this with any other existing LLMs. This is due to the fact our evaluation setting involves providing instructions to ChatGPT in a conversational manner, similar to the ones typically given to human evaluators. To the best of our knowledge, other than ChatGPT, there is no model that is currently publicly available for inference that can aptly understand these instructions to describe complex tasks and execute the tasks as instructed. Consequently, we only experiment with ChatGPT and report our findings accordingly.

6.4.1.2 Results

Paraphrasing: ChatGPT classified the correct and incorrect paraphrases with an accuracy score of 0.81. For the correct ones, precision and recall scores are 0.84 and 0.77 with an F1 score of 0.80 for 60 datapoints, and for the incorrect ones, precision and recall scores are 0.78 and 0.85 with an F1 score of 0.82 for the same amount of datapoints.

Question-answering: ChatGPT classified the correct and incorrect answers with an accuracy score of 0.89. For the correct answers, precision and recall scores are 0.81 and 0.88 with an F1 score of 0.85 for 34 datapoints, and for the incorrect answers, precision and recall scores are 0.94 and 0.89 with an F1 score of 0.91 for 65 datapoints.

Story-cloze test: For this task, ChatGPT was able to choose the correct fifth sentence with an accuracy score of 0.99 for a total datapoints of 157 short stories. Since it is an MCQ style question that requires identifying only the correct answer between two options, the individual scores (e.g., precision, recall) for each option are not relevant, hence we do not report them.

ECSGen: We conduct a series of analyses to understand ChatGPT's performance on the ECSGen task as the only dataset available for this task is relatively more complicated compared to the other evaluated tasks, which uses a 5-point Likert scale representing the degree of relevance of a suggestion to an emotion-cause. As explained in the previous section, 5-point Likert responses can be extremely subjective, particularly between the adjacent levels. Consequently, we analyze this using several techniques to understand ChatGPT's suitability as an automatic evaluator for the ECSGen task. The mean score of 5-point Likert responses from human evaluators and ChatGPT are 2.83 and 2.47 respectively, with a median of 3 and 2, and a standard deviation of 1.451 and 1.117 respectively. For the correlation test, we primarily use the Spearman coefficient (0.334) as

it fits the data type and distribution used in the study. However, we also report its parametric counterpart: the Pearson coefficient (0.33), as an additional correlation measure. Both correlation tests indicate a positive correlation between ChatGPT and human responses and both tests were significant with a p-value < 0.001.

We also report the distance between the Likert responses for each question to understand the percentages of the answers that varied differently. About 26.11% of suggestions had the exact same ranking by human evaluators and ChatGPT. ChatGPT and human evaluators' responses varied by only 1 Likert point for 40% of the suggestions. This indicates that more than 65% of the responses are an exact match or only varied by 1 Likert-point. For the remainder of the suggestions, 22.50% varied by 2 Likert points, 10.28% varied by 3, and only 1.11% varied by 4 Likert-point.

Additionally, we measure ChatGPT's effectiveness as an automatic evaluator for the ECSGen task by framing this analysis as a classification problem. However, since the responses are in a 5-point Likert scale that uses typical labels for intensity measurement, the accuracy score is expectedly low (0.26) for classification with five categories (i.e., 5-point Likert scale). Consequently, we perform further analysis by recoding the Likert scores. First, we recode this as a 3-category classification problem, where Likert score 1 is recoded as 1, 5 is recoded as 3, and the more ambiguous scores in between are recoded to 2. This expectedly increases the classification accuracy (0.55) compared to the 5-category approach. Next, we recode the original data again in 3-categories. However, this time, we recode 1 and 2 as 1, 4 and 5 as 3, and 3 as 2. This yields a similar classification accuracy (0.48) as the previous 3-category approach although slightly lower. We also analyze this from a binary classification perspective, for which we use two different approaches. First, we remove the datapoints with a Likert score of 3 from the original 5-point Likert data (i.e., 3 representing the mathematical mid-point in a 5-point Likert scale). We recode 1 and 2 as 1, and 4 and 5 as 2. This increases the classification accuracy further to 0.67. Lastly, we look at the agreement of human evaluators and ChatGPT for only the lowest and the highest scores: 1 and 5. This dramatically reduces the datapoints for analysis, but it provides a closer representation of a binary classification problem. The classification accuracy with this approach is significantly higher with a score of 0.91. The confusion matrices in Figure 1 provide a more detailed view of these various classification results.

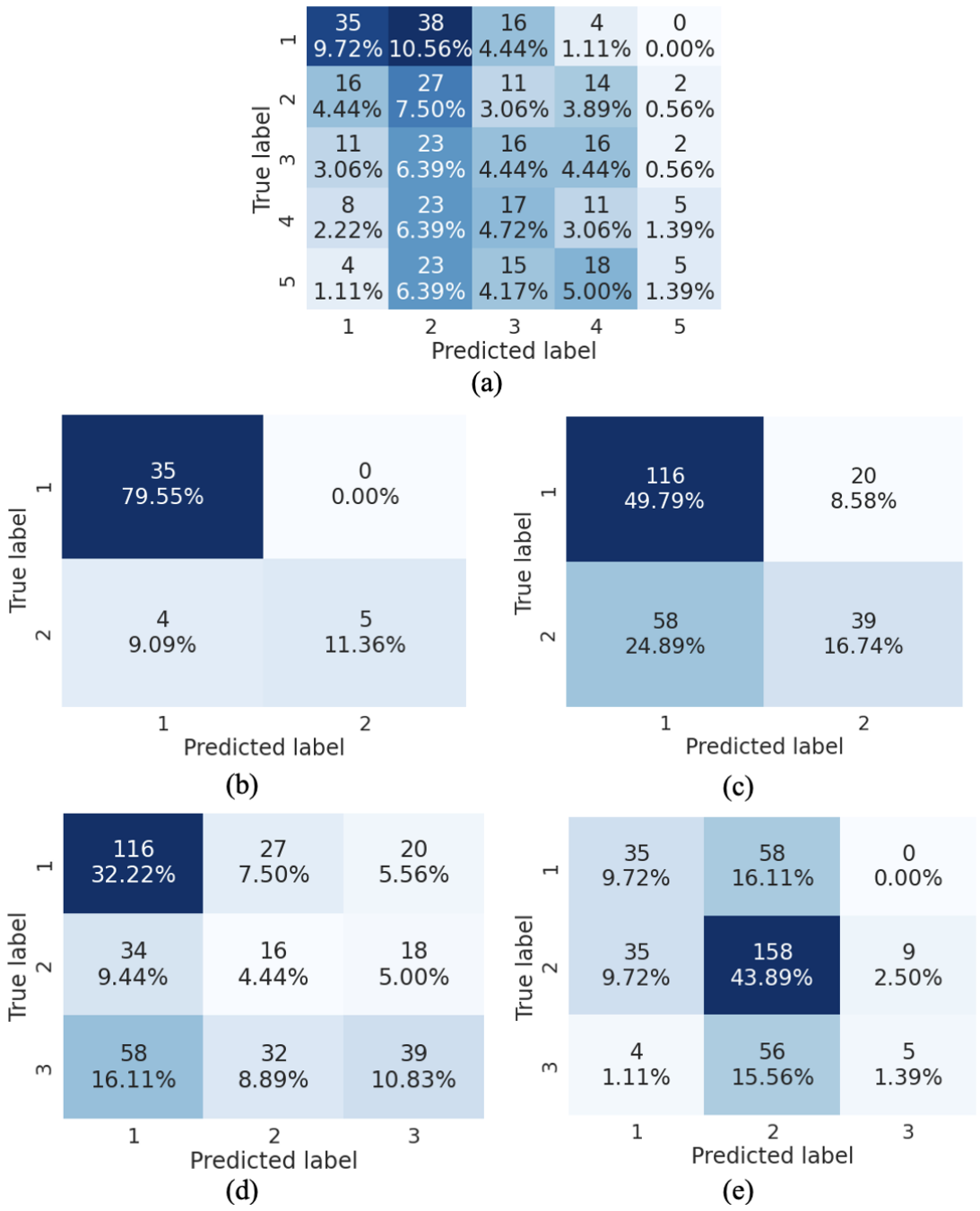

Figure 39: ECSGen confusion matrices: a) 5-class. b) binary with only labels 1 and 5. c) binary with label 3 removed and 1, 2 recoded to 1 and 4, 5 recoded to 2. d) 3-class with label 1 and 2 recoded as 1; 4 and 5 as 3; and 3 as 2. c) 3-class recoded 1 and 2 as 1; 4 and 5 as 3; and 3 as 2.

ECG: Similar to the Story-cloze test, for the ECG task, ChatGPT is tasked with MCQ-style questions where the requirement is to choose an emotion-cause from the two given options that

more appropriately explains the emotion expressed in a given statement. ChatGPT achieved an accuracy score of 0.98 for a total datapoints of 120.

Table 20 represents the summary of ChatGPT's performance in evaluating all the NLG tasks considered in this study.

Task	Analysis Method(s)	Score(s)
Paraphrasing	Classification Accuracy	0.81
Question Answering	Classification Accuracy	0.89
Story-cloze	Classification Accuracy Score (MCQ)	0.99
ECSGen	Spearman Coefficient, Pearson Coefficient	0.334, 0.33
	Classification Accuracy - Number of classes: 5, 3, 3*, 2, 2*	0.26, 0.55, 0.48*, 0.67, 0.91*
ECG	Classification Accuracy (MCQ)	0.98

Table 20: Summary of ChatGPT's performance as an evaluator for various NLG tasks. Asterisk (*) indicates an alternative recoding technique used (more details above).

6.5 Discussion

We have experimented with five NLG tasks to understand if ChatGPT, a recently released conversational LLM, can be an automatic evaluator for these tasks. Three of them are among the common NLG tasks and the remaining two have been recently introduced within the ECA domain. We observe that ChatGPT is able to understand all these tasks and conduct the evaluation for them with minimal instructions akin to human evaluators. In our opinion, this is a significant strength of Conversational LLMs like ChatGPT that makes them a good candidate to be automatic evaluators for many NLG tasks where researchers can quickly evaluate their models' performance for a given NLG task by providing simple instructions to ChatGPT. In addition, we leverage various techniques to evaluate ChatGPT's strength as an automatic evaluator for the given NLG tasks. These include framing the analysis as a classification problem with two or more categories, MCQ style questions, and Likert-scale ranking. We observe that ChatGPT is capable of understanding these diverse framings of problems with minimal instructions, which, we believe, reinforces its potential as an automatic evaluator for many NLG tasks.

We observe that ChatGPT achieves almost 100% accuracy for two tasks: ECG and Story-cloze test. For question-answering and paraphrasing it scored 0.89 and 0.81 respectively. We opine that

these consistent high scores represent ChatGPT's ability to be an effective evaluator for these NLG tasks. In our analysis, the ECSGen task's evaluation is different than the others experimented with in this study given the fact that its dataset comes with a 5-point Likert response from human evaluators which is prone to subjectivity. As explained earlier, we perform more analysis for this task to understand ChatGPT's suitability to be an effective evaluator for this task. Based on the classification accuracy that ChatGPT achieved in our analysis of the ECSGen task, particularly when framed as a binary problem, we suggest that ChatGPT can be considered a useful evaluator for this task as well. This is also reinforced by the positive score in the correlation tests between ChatGPT and human evaluators' ranking. In addition, we observe that the distance between the Likert responses from ChatGPT and human evaluators indicates a high correlation between them. Particularly, more than 65% of the responses had a Likert distance of 1 or 0, while only about 12% of the responses varied widely with a Likert distance of 3 or more. However, we believe that for ECSGen, a further study may be beneficial to confirm the findings of our study, especially with a binary ranking from human evaluators, which can be then compared against ChatGPT's response with more accuracy.

Based on our overall results, we suggest that conversational LLMs like ChatGPT can be effective automatic evaluators for the NLG tasks used in this study. We believe that this statement can potentially be generalized to many other similar NLG tasks. This is because, in our observation, ChatGPT is able to understand instructions for evaluating a variety of NLG tasks, some of which have been introduced very recently. In our opinion, this ability to understand how to evaluate a new NLG task from simple instructions, and then execute those instructions with decent performance are the two most important features of ChatGPT that can make it an effective automatic evaluator for many NLG tasks in the future. However, to use ChatGPT as an automatic evaluator for NLG tasks other than the ones used in this study, we recommend performing a small-scale evaluation with ChatGPT for that task. We expect that the evaluations we performed in this study can be an inspiration for such future experiments.

We observe that designing an appropriate prompt is crucial to help ChatGPT understand the task properly and respond accordingly. We recommend careful consideration while designing the prompts to use ChatGPT as an automatic evaluator for a given NLG task.

Lastly, since this is a newly suggested automatic evaluation technique, we expect this to mature over time with more successive contributions from the researcher community, especially with larger-scale studies. We expect this to initially complement existing evaluation techniques such as human evaluations. This may pave the way for this technique to gradually become one of the mainstream automatic evaluation techniques in the future for many NLG tasks, especially the ones that do not require specific domain knowledge. There is ongoing research to develop more conversational LLMs like ChatGPT [300]. When more such LLMs are available, preferably with enhanced performance, we believe researchers can leverage multiple conversational LLMs as automatic evaluators and report their findings accordingly for increasing the robustness of their analysis.

6.5.1 Guidelines for Researchers

We recommend transparency while using Conversational LLMs such as ChatGPT as an automatic evaluator for NLG tasks. Following are our general guidelines for this use-case:

- After the NLG task is complete, researchers can pick a small sample from the output (e.g., 10-20 datapoints, depending on the type of task).
- Researchers can use the Conversational LLM of choice to evaluate these sampled outputs.
- Optionally, researchers can then provide their own evaluation of these sampled outputs (or they can choose to ask human evaluators to do this).
- Then, these two evaluations of the sampled outputs, one by conversational LLM and another by the human can be presented in the research publication together in complete detail (i.e., evaluation of each datapoint in the sampled outputs, along with the input provided, and output generated). Alternatively, if human evaluation is not performed, the researchers can choose to only report the detailed evaluation from the Conversational LLM.
- This will provide the readers an understanding of the performance of the automatic evaluator which they can consider when interpreting the findings in a given study that includes automatic evaluation of NLG tasks using conversational LLM.

6.6 Limitations and Future Studies

There are several limitations of this study. The chief among them is the fact that we used a small number of samples in our experiments. As described above, this is because ChatGPT Research

Preview currently only allows access through a web browser and manually inputting a large amount of datapoint can be tedious and erroneous. It is also because we believe that such a sample is sufficient to get an indication of the general efficacy of ChatGPT as an evaluator of the NLG tasks discussed in this study. It is however a noteworthy limitation, and we acknowledge the need to investigate this further with larger samples.

We investigated ChatGPT's effectiveness as an automatic evaluator for several NLG tasks. However, there are many other NLG tasks that we have not experimented with. We have constrained this study to a small subset of NLG tasks to keep the experiments within the scope of this study. We acknowledge that our results cannot be generalized to every other NLG task, particularly the likes of machine translation, which we believe Conversational LLMs like ChatGPT are currently not able to effectively evaluate due to their limited knowledge of languages other than English. We encourage researchers to investigate the efficacy of ChatGPT and possible similar conversational LLMs as automatic evaluators for more NLG tasks.

While we encourage the usage of conversational LLMs such as ChatGPT for evaluating open-ended NLG tasks, evaluation of many tasks may require awareness of different additional factors such as cultural and geographical context. This may not be an issue for tasks such as paraphrasing or summarization, however, this can be an important factor for tasks such as ECG and ECSGen. For example, for the ECG task, which requires generation of a meaningful cause of an emotion expressed in a given text, there can be many answers, some can be more relevant than others based on different cultural context. For instance, different parts of the world have different public holidays that are reasons for celebration and joy. A cause for emotion related to these celebrations may be meaningful in one country but not the others. The same is true for the ECSGen task – the same suggestion to mitigate a particular negative emotion may not be appropriate all over the world. In the context of geographical and cultural differences, this is akin to the well-established concept of localization (i.e., the process of adapting a product or service to suit the language, culture, and specific requirements of a particular region, including the translation of content, modification of graphics, and compliance with local laws) [301]. We opine that there is opportunity to borrow from the existing concepts such as this one in order to enhance the conversational LLMs to have such awareness such as cultural and geographical context. This type of awareness within these LLMs can make them better evaluators of the open-ended NLG tasks

such as ECG and ECSGen. We believe this provides interesting future research opportunities regarding more effective usages of conversational LLMs in general.

The current public access of ChatGPT only allows interaction with the model without granting full access to it. In addition, the model is regularly updated with more training data. As a result, its response to one question may change over time. While we expect the response of ChatGPT to become more enhanced over time, we, however, suggest that the apparent varying nature of ChatGPT needs to be considered while interpreting the results we publish in this study.

6.7 Conclusions

We experimented with five NLG tasks to understand the performance of a recently released conversational LLM named ChatGPT as an automatic evaluator for these tasks. There are two ECA related tasks within these five tasks representing all currently available NLG tasks within the ECA domain. We observe that ChatGPT is able to understand how to execute these tasks with simple instructions akin to the ones typically given to the human participants who evaluate these tasks. We believe that this capability to understand NLG tasks from simple instructions along with the results from our experiments suggest that ChatGPT and possible future and more advanced variants of such conversational LLM can be effective automatic evaluators for the NLG tasks explored in this study. We opine that with more experiments, this finding can be potentially generalized to many other similar NLG tasks, especially the tasks that do not require domain specific knowledge.

Chapter 7: Overall Conclusions

In this thesis, we have conducted multiple investigations on several NLP tasks related to human emotion and sentiment. We devised two successive techniques to perform multi-class Sentiment Analysis using only 50 labeled training data per class. Extending the focus beyond the classification of sentiment towards the cause of the emotion, we introduced two novel generative NLP tasks (i.e., NLG tasks) in the domain of Emotion-Cause Analysis (ECA) and developed technical solutions to perform these tasks. Additionally, building upon our experience with the NLG task evaluation, we proposed a new automatic evaluation method for open-ended NLG tasks and conducted experiments to demonstrate its viability. For all our investigations on these various NLP tasks, primarily related to human emotion and sentiment, we devised technical solutions that could operate with little to no labeled data. This is largely facilitated by leveraging Transformer-based large language models (LLMs) as a core element in our proposed architectures. We provide a summary of each investigation we conducted in this thesis below:

- We developed SG-Elect technical architecture to perform multi-class Sentiment Analysis. Our experiments showed that SG-Elect achieved significantly higher performance (i.e., F1 macro, and F1 weighted average) in multi-class Sentiment Analysis compared to its baseline. Building upon SG-Elect, we developed GAN-BElectra, which significantly reduced the architecture complexity and required training steps compared to SG-Elect, without sacrificing the performance gain achieved by its predecessor. It is noteworthy that GAN-BElectra in fact achieved a small gain in performance (arithmetically) consistently across all datasets we evaluated though the gain did not reach statistical significance.
- We proposed the first ever NLG task within the ECA domain named Emotion-Cause Generation (ECG). The goal of this task is to generate a meaningful cause for an emotion expressed in a given text. We positioned this task as an infilling task and established its viability as an NLP task through technical demonstration. We customized an existing infilling architecture and repurposed an available ECA dataset (containing a total of 820 labeled datapoints for training, validation, and inference) to perform the demonstration.
- We designed a second novel NLG task in the ECA domain named Emotion-Cause mitigating Suggestion Generation (ECSGen). The objective of this task is to generate relevant suggestions to mitigate a negative emotion expressed in a given text statement. We proposed iZen, a zero-

shot framework to perform this without requiring any labeled data or new training steps. Our experiments suggested that iZen could generate suggestions with decent relevancy as part of the ECSGen task. We also curated our own dataset to perform this task.

- Lastly, we proposed a novel automatic evaluation method for open-ended NLG tasks leveraging Conversational Large Language Models. We performed several experiments with ChatGPT which was the most advanced conversational LLM available at the time of this investigation. These experiments suggested the promising ability of conversational LLMs as effective automatic evaluators for open-ended NLG tasks. Additionally, we provided guidelines for the researchers on how to use conversational LLMs in their study for such use-case.

7.1 Implications of the Contributions

The implications of the contributions of this research are manifold. We discuss them below:

- Sentiment Analysis is a highly context-dependent task where the same set of words can express different sentiments in different settings and domains. Consequently, it warrants domain-specific labeled data to train machine learning models that can perform the task with decent accuracy, which can be expensive. We proposed two performant techniques for the multi-class Sentiment Analysis task which only required 50 labeled datapoints per class. Thus, we opine that these two techniques are important contributions in the specific problem space of multi-class Sentiment Analysis with limited labeled data.
- Despite sheer advancements in generative machine learning technologies in recent years, the ECA domain only had NLU tasks to date. This thesis introduced two novel NLG tasks (ECG, ECSGen) in this domain with accompanying technical solutions. The solution for the ECG task utilized a small existing dataset containing only 820 labeled datapoints for training, validation, and inference combined. The technical architecture to perform ECSGen, named iZen, did not require any labeled data or new training steps. Additionally, we published a new dataset for the ECSGen task. We believe all these contributions significantly enrich the ECA research domain and pave the way for more similar and innovative investigations in this area.
- Evaluation of NLG tasks typically requires human participants in the evaluation process, particularly for open-ended generative tasks such as summarizing, and poetry generation. This is a time-consuming process for this rapidly advancing field. We proposed that Conversational

LLMs like ChatGPT can be effective automatic evaluators for open-ended NLG tasks. We performed several experiments that demonstrated the viability of this proposal. We provided guidelines for the researchers to effectively use conversational LLM in their study which we expect to create more transparency in reporting of results generated through such conversational LLMs. We believe that the proper utilization of this proposed automatic evaluation method for open-ended NLG tasks can significantly benefit this vast research area, potentially accelerating the velocity of innovation in this area.

In summary, this thesis makes important contributions to the NLP field on several fronts, particularly NLP tasks related to human emotion: by introducing generative NLP tasks in the ECA domain, by devising technical solutions to perform an existing classification task (i.e., multi-class Sentiment Analysis) and two new generative tasks (i.e., ECG, ECSGen) in limited to no labeled data settings, and by proposing a novel method for automatic evaluation of open-ended NLG tasks. Our investigations also demonstrated the general capability of Transformer-based LLMs in performing various tasks in limited to no labeled data situations, spanning both NLU and NLG subdivisions of NLP, and for both existing and novel tasks. This reinforces the ongoing utilization of LLMs in a variety of NLP tasks in different domains and invites further investigations into the potential utility of these LLMs in more NLP tasks.

7.2 Future Work

Research presented in this thesis activates several avenues for potential future work. We discuss them below:

- Both technical solutions (SG-Elect and GAN-BElectra) that we proposed for multi-class Sentiment Analysis with limited labeled data required multi-step training. This makes the overall training more tedious, time-consuming, and error prone. A worthwhile future work in this area is to devise architecture with unified training which can achieve similar or higher classification accuracy. Another useful research in this area could be taking the zero-shot approach where no labeled training data is required for multi-class Sentiment Analysis, while still attaining equivalent or greater classification accuracy. Lastly, similar inquiries within the limited or no labeled data circumstances can be extended towards various subtasks of Sentiment Analysis such as aspect-oriented Sentiment Analysis.

- We introduced generative NLP tasks within the ECA domain. This is expected to create potential future research opportunities on various fronts. One direction could be devising more sophisticated and performant technical solutions to perform the novel tasks proposed in this thesis. We expect the ECSGen dataset that we publish as part of our research to be beneficial in accelerating more research around this task. Secondly, we believe that our research can be a catalyst for creating more new generative tasks within the ECA domain. Extending on this front, there is an opportunity to combine the existing ECA related NLU tasks with the proposed NLG tasks. For example, a solution can be devised that can perform ECSP (Emotion Cause-Span Pair Extraction) on a given text to identify the emotion expressed in a given sentence along with extracting the cause-span related to that emotion. Subsequently, it can perform ESCGen to generate a relevant suggestion to mitigate that emotion. In our thesis, the ECSGen task only took input statements with negative emotions in order to generate a suggestion to mitigate that. A worthwhile future work can be to extend the ECSGen task so that it can take a statement with any kind of emotion, and based on the emotion, whether given or classified during the inference time, a solution generates a suggestion if the emotion is negative or generates a relevant comment if the emotion is positive.
- Lastly, we proposed using conversational LLMs for the automatic evaluation of open-ended NLG tasks. We performed a few small-scale experiments to prove our hypothesis. An expected future work can be performing more rigorous experiments with more NLG tasks. In addition, we only evaluated the performance of one conversational LLM: ChatGPT. As more such technologies become available, the outcome of this research can be more firmly established by experimenting with more of these technologies. We also believe that our proposal creates an opportunity to develop more structured guidelines for using these automatic evaluators so that it can be widely adopted following a common procedure.

Appendices

Appendix A

A.1 Guidelines for participants on how to respond to the survey questions for the ECSGen Task

An example to show how to answer the survey questions:

Each question has two parts:

Part 1:
Includes a "statement" (it expresses a negative emotion and provides a cause of that emotion) and three "suggestions" to address the "cause of the negative emotion" expressed in the "statement".

Statement: I am sad because I failed an exam yesterday.
Suggestion **A**: You should prepare well before the exam.
Suggestion **B**: You can sell your car.
Suggestion **C**: Then you should look at the test results.

For this question, we can rank the relevancy of each suggestion in the following way:
*Suggestion A is **VERY RELEVANT***
*Suggestion B is **NOT RELEVANT AT ALL***
*Suggestion C is **SLIGHTLY RELEVANT***

Part 2:
Includes a question about the "statement" itself:
Does the **Statement** mentioned above contain enough information to provide a relevant suggestion?

*For this statement, it seems that the "statement" includes enough information to provide a simple suggestion. So we can answer this as - **YES**!*

In case there is a "statement" that does not contain enough information to provide a suggestion, then the answer would be "NO". For example, "I am feeling frustrated because of this". This statement does not contain enough information to provide a simple suggestion.

Hope this example helps! Let's start!

A.2 A Sample Questionnaire from the Survey for the ECSGen Task

Statement: I am feeling sad because no one nominated
Suggestion **A**: Go onto the voting threads
Suggestion **B**: Pm me
Suggestion **C**: You should be the one to do so
Are these suggestions relevant to the above **Statement**? Please rate each suggestion's relevancy below:

	Not at all relevant	Slightly relevant	Somewhat relevant	Moderately relevant	Very relevant
Suggestion A	○	○	○	○	○
Suggestion B	○	○	○	○	○
Suggestion C	○	○	○	○	○

Does the **Statement** mentioned above contains **enough information** to provide a relevant suggestion?

- Yes
- Maybe
- No

References

[1] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: state of the art, current trends and challenges," *Multimed Tools Appl*, vol. 82, no. 3, pp. 3713–3744, Jan. 2022, doi: 10.1007/S11042-022-13428-4/FIGURES/3.

[2] "Guidelines for PhD Integrated Thesis Format | Recreation and Leisure Studies | University of Waterloo." https://uwaterloo.ca/recreation-and-leisure-studies/current-graduate-students/thesis-information/doctoral-thesis/guidelines-phd-integrated-thesis-format (accessed Mar. 08, 2023).

[3] C. Wohlin, "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering," 2014, doi: 10.1145/2601248.2601268.

[4] "Google Scholar." https://scholar.google.ca/ (accessed Jan. 24, 2021).

[5] "Omni - Carleton University Library." https://ocul-crl.primo.exlibrisgroup.com/discovery/search?vid=01OCUL_CRL:CRL_DEFAULT (accessed Mar. 08, 2023).

[6] K. R. Scherer, "What are emotions? And how can they be measured?," *http://dx.doi.org/10.1177/0539018405058216*, vol. 44, no. 4, pp. 695–729, Dec. 2005, doi: 10.1177/0539018405058216.

[7] C. J. Fillmore, J. Ruppenhofer, and A. Wright, "FRAMENET IN ACTION: THE CASE OF ATTACHING".

[8] A. Ortony, G. L. Clore, and A. Collins, "The Cognitive structure of emotions cambridge," *UK: Cambridge University Press9*, 1988.

[9] D. Ghazi, D. Inkpen, and S. Szpakowicz, "Detecting Emotion Stimuli in Emotion-Bearing Sentences," in *Computational Linguistics and Intelligent Text Processing*, Springer International Publishing, 2015, pp. 152–165.

[10] W. V. F. and P. E. PAUL EKMAN, *Emotion in the Human Face*. Elsevier, 1972. doi: 10.1016/c2013-0-02458-9.

[11] R. Plutchik, "A psychoevolutionary theory of emotions," *Social Science Information*, vol. 21, no. 4–5, pp. 529–553, Feb. 1982, doi: 10.1177/053901882021004003.

[12] B. Pang and L. Lee, “Opinion Mining and Sentiment Analysis,” *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008, doi: 10.1561/1500000011.

[13] J. J. Gross, “Emotion regulation: Conceptual and empirical foundations.,” in *Handbook of emotion regulation*, 2014. Accessed: Mar. 12, 2023. [Online]. Available: https://psycnet.apa.org/record/2013-44085-001

[14] C. E. Osgood, G. J. Suci, and P. H. Tannenbaum, *The measurement of meaning*, no. 47. University of Illinois press, 1957.

[15] B. Liu, “Sentiment analysis and opinion mining,” *Synthesis Lectures on Human Language Technologies*, vol. 5, no. 1, pp. 1–184, May 2012, doi: 10.2200/S00416ED1V01Y201204HLT016.

[16] “Cambridge Dictionary | English Dictionary, Translations & Thesaurus.” https://dictionary.cambridge.org/ (accessed Apr. 25, 2021).

[17] T. Nasukawa and J. Yi, “Sentiment analysis: Capturing favorability using natural language processing,” in *Proceedings of the 2nd International Conference on Knowledge Capture, K-CAP 2003*, New York, New York, USA: Association for Computing Machinery, Inc, Oct. 2003, pp. 70–77. doi: 10.1145/945645.945658.

[18] J. M. Wiebe, “Learning Subjective Adjectives from Corpora,” 2000.

[19] W. Medhat, A. Hassan, and H. Korashy, “Sentiment analysis algorithms and applications: A survey,” *Ain Shams Engineering Journal*, vol. 5, no. 4, pp. 1093–1113, Dec. 2014, doi: 10.1016/j.asej.2014.04.011.

[20] N. C. Dang, M. N. Moreno-García, and F. De la Prieta, “Sentiment analysis based on deep learning: A comparative study,” *Electronics (Switzerland)*, vol. 9, no. 3, Mar. 2020, doi: 10.3390/electronics9030483.

[21] B. Pang and L. Lee, “Opinion mining and sentiment analysis,” *Foundations and Trends in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008, doi: 10.1561/1500000011.

[22] A. Alsaeedi and M. Z. Khan, “A Study on Sentiment Analysis Techniques of Twitter Data,” *Article in International Journal of Advanced Computer Science and Applications*, vol. 10, no. 2, 2019, doi: 10.14569/IJACSA.2019.0100248.

[23] K. Chakraborty, S. Bhattacharyya, R. Bag, and A. A. Hassanien, “Sentiment Analysis on a Set of Movie Reviews Using Deep Learning Techniques,” in *Social Network Analytics*, Elsevier, 2019, pp. 127–147. doi: 10.1016/b978-0-12-815458-8.00007-4.

[24] M. E. Mowlaei, M. Saniee Abadeh, and H. Keshavarz, “Aspect-based sentiment analysis using adaptive aspect-based lexicons,” *Expert Syst Appl*, vol. 148, p. 113234, 2020, doi: 10.1016/j.eswa.2020.113234.

[25] L. Yue, W. Chen, X. Li, W. Zuo, and M. Yin, “A survey of sentiment analysis in social media,” *Knowl Inf Syst*, vol. 60, pp. 617–663, 2019, doi: 10.1007/s10115-018-1236-4.

[26] P. D. Turney, “Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews.”

[27] M. Hu and B. Liu, “Mining and Summarizing Customer Reviews,” 2004.

[28] A. Esuli and F. Sebastiani, “SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining.”

[29] S. M. Mohammad, S. Kiritchenko, and X. Zhu, “NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets,” 2013.

[30] “General Inquirer Categories.” http://www.wjh.harvard.edu/~inquirer/homecat.htm (accessed May 24, 2021).

[31] “Lasswell Dictionary Description.” http://www.wjh.harvard.edu/~inquirer/lasswell.htm (accessed May 24, 2021).

[32] “WordNet | A Lexical Database for English.” https://wordnet.princeton.edu/ (accessed May 02, 2021).

[33] M. Thelwall, K. Buckley, and G. Paltoglou, “Sentiment Strength Detection for the Social Web 1,” 2011.

[34] R. Ortega, A. Fonseca, A. P. Lumumba S/, N. Santiago De Cuba, and C. Yoan Gutiérrez, “SSA-UO: Unsupervised Twitter Sentiment Analysis.”

[35] P. Nakov, Z. Kozareva, A. Ritter, S. Rosenthal, V. Stoyanov, and T. Wilson, “SemEval-2013 Task 2: Sentiment Analysis in Twitter,” 2013.

[36] H. Saif, Y. He, M. Fernandez, and H. Alani, “Contextual Semantics for Sentiment Analysis of Twitter,” 2015.

[37] S.-M. Kim and E. Hovy, “Determining the Sentiment of Opinions.”

[38] S. Feng, R. Bose, and Y. Choi, “Learning General Connotation of Words using Graph-based Algorithms.”

[39] “PageRank - Wikipedia.” https://en.wikipedia.org/wiki/PageRank (accessed May 24, 2021).

[40] C. Strapparava and R. Mihalcea, “SemEval-2007 Task 14: Affective Text,” 2007. doi: 10.5555/1621474.1621487.

[41] M. D. P. Salas-Zárate, J. Medina-Moreira, K. Lagos-Ortiz, H. Luna-Aveiga, M. Á. Rodríguez-García, and R. Valencia-García, “Sentiment Analysis on Tweets about Diabetes: An Aspect-Level Approach,” *Comput Math Methods Med*, vol. 2017, 2017, doi: 10.1155/2017/5140631.

[42] “n-gram - Wikipedia.” https://en.wikipedia.org/wiki/N-gram (accessed May 24, 2021).

[43] A. Go, R. Bhayani, and L. Huang, “Twitter Sentiment Classification using Distant Supervision.”

[44] J. Read, “Using Emoticons to reduce Dependency in Machine Learning Techniques for Sentiment Classification,” 2005.

[45] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up? Sentiment Classification using Machine Learning Techniques,” EMNLP, 2002.

[46] A. Pak and P. Paroubek, “Twitter as a Corpus for Sentiment Analysis and Opinion Mining.”

[47] “Conditional random field - Wikipedia.” https://en.wikipedia.org/wiki/Conditional_random_field (accessed May 25, 2021).

[48] L. Barbosa and J. Feng, “Robust Sentiment Detection on Twitter from Biased and Noisy Data,” 2010.

[49] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao, “Target-dependent Twitter Sentiment Classification.”

[50] K. Korovkinas, P. Danėnas, G. Garšva, P. Dan, and G. Gař Sva, “SVM and k-Means Hybrid Method for Textual Data Sentiment Analysis,” *Baltic J. Modern Computing*, vol. 7, no. 1, pp. 47–60, 2019, doi: 10.22364/bjmc.2018.7.1.04.

[51] K. Korovkinas, P. Danėnas, and G. Garšva, “SVM Accuracy and Training Speed Trade-Off in Sentiment Analysis Tasks,” in *Communications in Computer and Information Science*, Springer Verlag, Oct. 2018, pp. 227–239. doi: 10.1007/978-3-319-99972-2_18.

[52] R. K. BANIA, “COVID-19 Public Tweets Sentiment Analysis using TF-IDF and Inductive Learning Models,” no. 2, 2020.

[53] “Parametric and Nonparametric Machine Learning Algorithms.” https://machinelearningmastery.com/parametric-and-nonparametric-machine-learning-algorithms/ (accessed May 25, 2021).

[54] “COVID-19 pandemic - Wikipedia.” https://en.wikipedia.org/wiki/COVID-19_pandemic (accessed May 25, 2021).

[55] S. Riaz, M. Fatima, M. Kamran, and M. W. Nisar, “Opinion mining on large scale data using sentiment analysis and k-means clustering,” *Cluster Comput*, vol. 22, no. 3, pp. 7149–7164, May 2019, doi: 10.1007/s10586-017-1077-z.

[56] V. Iosifidis and E. Ntoutsi, “Large Scale Sentiment Learning with Limited Labels,” vol. 10, 2017, doi: 10.1145/3097983.3098159.

[57] J. Khan and Y. K. Lee, “LeSSA: A unified framework based on lexicons and semi-supervised learning approaches for textual sentiment classification,” *Applied Sciences (Switzerland)*, vol. 9, no. 24, Dec. 2019, doi: 10.3390/app9245562.

[58] D. Tang, B. Qin, and T. Liu, “Deep learning for sentiment analysis: successful approaches and future challenges,” *WIREs Data Mining Knowl Discov*, vol. 5, pp. 292–303, 2015, doi: 10.1002/widm.1171.

[59] X. Zhang and X. Zheng, “Comparison of text sentiment analysis based on machine learning,” in *Proceedings - 15th International Symposium on Parallel and Distributed Computing, ISPDC 2016*, Institute of Electrical and Electronics Engineers Inc., Apr. 2017, pp. 230–233. doi: 10.1109/ISPDC.2016.39.

[60] “Extreme learning machine - Wikipedia.” https://en.wikipedia.org/wiki/Extreme_learning_machine (accessed May 25, 2021).

[61] S. Sohangir, D. Wang, A. Pomeranets, and T. M. Khoshgoftaar, “Big Data: Deep Learning for financial sentiment analysis,” *J Big Data*, doi: 10.1186/s40537-017-0111-6.

[62] J. Qian, Z. Niu, and C. Shi, “Sentiment analysis model on weather related tweets with deep neural network,” in *ACM International Conference Proceeding Series*, New York, New York, USA: Association for Computing Machinery, Feb. 2018, pp. 31–35. doi: 10.1145/3195106.3195111.

[63] D.-H. Pham and A.-C. Le, “Learning Multiple Layers of Knowledge Representation for Aspect Based Sentiment Analysis.”

[64] Q. T. Ain *et al.*, “Sentiment Analysis Using Deep Learning Techniques: A Review,” 2017.

[65] Y. Gao, W. Rong, Y. Shen, and Z. Xiong, “Convolutional Neural Network based sentiment analysis using Adaboost combination,” in *Proceedings of the International Joint Conference on Neural Networks*, Institute of Electrical and Electronics Engineers Inc., Oct. 2016, pp. 1333–1338. doi: 10.1109/IJCNN.2016.7727352.

[66] “Word2vec - Wikipedia.” https://en.wikipedia.org/wiki/Word2vec (accessed Dec. 15, 2020).

[67] U. Gupta, A. Chatterjee, R. Srikanth, and P. Agrawal, “A Sentiment-and-Semantics-Based Approach for Emotion Detection in Textual Conversations,” Neu, 2017.

[68] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, “Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification *,” Association for Computational Linguistics.

[69] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global Vectors for Word Representation.” Accessed: May 25, 2021. [Online]. Available: http://nlp.

[70] L. Li, T. T. Goh, and D. Jin, “How textual quality of online reviews affect classification performance: a case of deep learning sentiment analysis,” *Neural Comput Appl*, vol. 32, no. 9, pp. 4387–4415, May 2020, doi: 10.1007/s00521-018-3865-7.

[71] S. Jebbara and P. Cimiano, “Zero-Shot Cross-Lingual Opinion Target Extraction.”

[72] O. Araque, I. Corcuera-Platas, J. F. Sánchez-Rada, and C. A. Iglesias, "Enhancing deep learning sentiment analysis with ensemble techniques in social applications," *Expert Syst Appl*, vol. 77, pp. 236–246, Jul. 2017, doi: 10.1016/j.eswa.2017.02.002.

[73] M. E. Basiri, S. Nemati, M. Abdar, E. Cambria, and U. R. Acharya, "ABCDM: An Attention-based Bidirectional CNN-RNN Deep Model for sentiment analysis," *Future Generation Computer Systems*, vol. 115, pp. 279–294, Feb. 2021, doi: 10.1016/j.future.2020.08.005.

[74] L. Gui, D. Wu, R. Xu, Q. Lu, and Y. Zhou, "Event-Driven Emotion Cause Extraction with Corpus Construction," pp. 1639–1649, Accessed: Jan. 23, 2022. [Online]. Available: http://hlt.hitsz.edu.cn/?page

[75] R. Xia and Z. Ding, "Emotion-Cause Pair Extraction: A New Task to Emotion Analysis in Texts," Jun. 2019, [Online]. Available: http://arxiv.org/abs/1906.01267

[76] H. Bi and P. Liu, "ECSP: A New Task for Emotion-Cause Span-Pair Extraction and Classification," Mar. 2020, [Online]. Available: http://arxiv.org/abs/2003.03507

[77] S. Yat, M. Lee, Y. Chen, and C.-R. Huang, "A Text-driven Rule-based System for Emotion Cause Detection," in *NAACL HLT*, 2010, pp. 45–53. [Online]. Available: http://dbo.sinica.edu.tw/SinicaCorpus/

[78] W. Li and H. Xu, "Text-based emotion classification using emotion cause extraction," *Expert Syst Appl*, vol. 41, no. 4 PART 2, pp. 1742–1749, 2014, doi: 10.1016/J.ESWA.2013.08.073.

[79] K. Gao, H. Xu, and J. Wang, "A rule-based approach to emotion cause detection for Chinese micro-blogs," *Expert Syst Appl*, vol. 42, no. 9, pp. 4517–4528, Jun. 2015, doi: 10.1016/J.ESWA.2015.01.064.

[80] K. Gao, H. Xu, and J. Wang, "Emotion cause detection for Chinese micro-blogs based on ECOCC model," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9078, pp. 3–14, 2015, doi: 10.1007/978-3-319-18032-8_1/COVER.

[81] Y. Chen, S. Yat Mei Lee, S. Li, and C.-R. Huang, "Emotion Cause Detection with Linguistic Constructions," pp. 179–187, 2010.

[82] J. Lafferty, A. McCallum, and F. Pereira, “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data,” *Departmental Papers (CIS)*, Jun. 2001, Accessed: Feb. 26, 2023. [Online]. Available: https://repository.upenn.edu/cis_papers/159

[83] S. Yada, K. Ikeda, K. Hoashi, and K. Kageura, “A Bootstrap Method for Automatic Rule Acquisition on Emotion Cause Extraction,” *IEEE International Conference on Data Mining Workshops, ICDMW*, vol. 2017-November, pp. 414–421, Dec. 2017, doi: 10.1109/ICDMW.2017.60.

[84] L. Gui, R. Xu, Q. Lu, D. Wu, and Y. Zhou, “Emotion cause extraction, a challenging task with corpus construction,” *Communications in Computer and Information Science*, vol. 669, pp. 98–109, 2016, doi: 10.1007/978-981-10-2993-6_8/COVER.

[85] R. Xu, J. Hu, Q. Lu, D. Wu, and L. Gui, “An ensemble approach for emotion cause detection with event extraction and multi-kernel SVMs,” *Tsinghua Sci Technol*, vol. 22, no. 6, pp. 646–659, Dec. 2017, doi: 10.23919/TST.2017.8195347.

[86] X. Cheng, Y. Chen, B. Cheng, S. Li, and G. Zhou, “An Emotion Cause Corpus for Chinese Microblogs with Multiple-User Structures,” *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, vol. 17, no. 1, Nov. 2017, doi: 10.1145/3132684.

[87] L. Gui, J. Hu, Y. He, R. Xu, Q. Lu, and J. Du, “A Question Answering Approach for Emotion Cause Extraction,” *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pp. 1593–1602, 2017, doi: 10.18653/V1/D17-1167.

[88] Z. Ding, H. He, M. Zhang, and R. Xia, “From Independent Prediction to Reordered Prediction: Integrating Relative Position and Global Label Information to Emotion Cause Identification,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 6343–6350, Jul. 2019, doi: 10.1609/AAAI.V33I01.33016343.

[89] R. Xia, M. Zhang, and Z. Ding, “RTHN: A RNN-Transformer Hierarchical Network for Emotion Cause Extraction,” Jun. 2019, [Online]. Available: http://arxiv.org/abs/1906.01236

[90] R. Xia and Z. Ding, “Emotion-Cause Pair Extraction: A New Task to Emotion Analysis in Texts,” Jun. 2019, [Online]. Available: http://arxiv.org/abs/1906.01267

[91] P. Wei, J. Zhao, and W. Mao, “Effective Inter-Clause Modeling for End-to-End Emotion-Cause Pair Extraction.”

[92] C. Yuan, C. Fan, J. Bao, and R. Xu, “Emotion-Cause Pair Extraction as Sequence Labeling Based on A Novel Tagging Scheme.” Accessed: Feb. 27, 2021. [Online]. Available: https://github.com/huggingface/

[93] Z. Cheng, Z. Jiang, Y. Yin, N. Li, and Q. Gu, “A Unified Target-Oriented Sequence-to-Sequence Model for Emotion-Cause Pair Extraction,” *IEEE/ACM Trans Audio Speech Lang Process*, vol. 29, pp. 2779–2791, 2021, doi: 10.1109/TASLP.2021.3102194.

[94] Q. Sun, Y. Yin, and H. Yu, “A Dual-Questioning Attention Network for Emotion-Cause Pair Extraction with Context Awareness,” Apr. 2021, [Online]. Available: http://arxiv.org/abs/2104.07221

[95] A. Vaswani *et al.*, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, Neural information processing systems foundation, Jun. 2017, pp. 5999–6009. Accessed: May 02, 2021. [Online]. Available: https://arxiv.org/abs/1706.03762v5

[96] C. Fan, C. Yuan, L. Gui, Y. Zhang, and R. Xu, “Multi-Task Sequence Tagging for Emotion-Cause Pair Extraction Via Tag Distribution Refinement,” *IEEE/ACM Trans Audio Speech Lang Process*, vol. 29, pp. 2339–2350, 2021, doi: 10.1109/TASLP.2021.3089837.

[97] M. Stern, J. Andreas, and D. Klein, “A Minimal Span-Based Neural Constituency Parser,” *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, vol. 1, pp. 818–827, May 2017, doi: 10.48550/arxiv.1705.03919.

[98] K. Lee, L. He, M. Lewis, and L. Zettlemoyer, “End-to-end Neural Coreference Resolution,” *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pp. 188–197, Jul. 2017, doi: 10.48550/arxiv.1707.07045.

[99] “tf–idf - Wikipedia.” https://en.wikipedia.org/wiki/Tf–idf (accessed Dec. 15, 2020).

[100] “Word embeddings | TensorFlow Core.” https://www.tensorflow.org/tutorials/text/word_embeddings (accessed May 01, 2021).

[101] J. Landthaler *et al.*, “Extending Thesauri Using Word Embeddings and the Intersection Method SaToS-Software aided analysis of Terms of Services View project Extending Thesauri Using Word Embeddings and the Intersection Method,” 2017.

[102] “Shallow Neural Networks. In this post, I have explained what… | by Rochak Agrawal | Towards Data Science.” https://towardsdatascience.com/shallow-neural-networks-23594aa97a5 (accessed May 24, 2021).

[103] “Deep learning vs. machine learning - Azure Machine Learning | Microsoft Docs.” https://docs.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning (accessed May 02, 2021).

[104] “Hyperparameter optimization - Wikipedia.” https://en.wikipedia.org/wiki/Hyperparameter_optimization (accessed May 24, 2021).

[105] “Difference Between Supervised, Unsupervised, & Reinforcement Learning | NVIDIA Blog.” https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/ (accessed Dec. 15, 2020).

[106] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-supervised learning*. Cambridge, Mass.: MIT Press, 2006.

[107] N. Li, C.-Y. Chow, and J.-D. Zhang, “SEML: A Semi-Supervised Multi-Task Learning Framework for Aspect-Based Sentiment Analysis,” *IEEE Access*, vol. 8, pp. 189287–189297, Oct. 2020, doi: 10.1109/access.2020.3031665.

[108] A. Nöu, “Logistic Regression versus Support Vector Machines Matematiska institutionen,” p. 42, 2018.

[109] V. V. Corinna Cortes, “Support-Vector Networks,” *Machine Leaming*, vol. 20, pp. 273–297, 1995, doi: 10.1109/64.163674.

[110] E. W. Weisstein, “Maximum Entropy Method”.

[111] V. Narayanan, I. Arora, and A. Bhatia, “Fast and accurate sentiment classification using an enhanced Naive Bayes model.”

[112] “Understanding Deep Learning: DNN, RNN, LSTM, CNN and R-CNN | by SPRH LABS | Medium.” https://medium.com/@sprhlabs/understanding-deep-learning-dnn-rnn-lstm-cnn-and-r-cnn-6602ed94dbff (accessed May 22, 2021).

[113] “Activation function - Wikipedia.” https://en.wikipedia.org/wiki/Activation_function (accessed May 22, 2021).

[114] “Training an Artificial Neural Network.” http://www2.psych.utoronto.ca/users/reingold/courses/ai/cache/neural3.html (accessed May 20, 2021).

[115] “Overview of GAN Structure | Generative Adversarial Networks.” https://developers.google.com/machine-learning/gan/gan_structure (accessed May 20, 2021).

[116] “Transfer learning - Wikipedia.” https://en.wikipedia.org/wiki/Transfer_learning (accessed May 02, 2021).

[117] D. Wang *et al.*, “Comprehensive eye diagram analysis: A transfer learning approach,” *IEEE Photonics J*, vol. 11, no. 6, Dec. 2019, doi: 10.1109/JPHOT.2019.2947705.

[118] D. Bahdanau, K. H. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Sep. 2014, doi: 10.48550/arxiv.1409.0473.

[119] T. Wolf *et al.*, “Transformers: State-of-the-art natural language processing,” *arXiv*. arXiv, pp. 38–45, Oct. 08, 2019. doi: 10.18653/v1/2020.emnlp-demos.6.

[120] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 4171–4186, Oct. 2018, Accessed: May 02, 2021. [Online]. Available: http://arxiv.org/abs/1810.04805

[121] “GPT-2 - Wikipedia.” https://en.wikipedia.org/wiki/GPT-2 (accessed May 02, 2021).

[122] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int J Comput Vis*, vol. 115, no. 3, pp. 211–252, Dec. 2015, doi: 10.1007/S11263-015-0816-Y/FIGURES/16.

[123] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, Jan. 2013, doi: 10.48550/arxiv.1301.3781.

[124] A. M. Dai and Q. v. Le, “Semi-supervised Sequence Learning,” *Adv Neural Inf Process Syst*, vol. 28, 2015, Accessed: Feb. 26, 2023. [Online]. Available: http://ai.Stanford.edu/amaas/data/sentiment/index.html

[125] J. Howard and S. Ruder, “Universal Language Model Fine-tuning for Text Classification,” *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, vol. 1, pp. 328–339, Jan. 2018, doi: 10.48550/arxiv.1801.06146.

[126] M. E. Peters *et al.*, “Deep Contextualized Word Representations,” *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 2227–2237, 2018, doi: 10.18653/V1/N18-1202.

[127] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving Language Understanding by Generative Pre-Training,” *OpenAI*, 2018, Accessed: Jan. 08, 2023. [Online]. Available: https://gluebenchmark.com/leaderboard

[128] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners,” *OpenAI blog,* vol. 1(8), no. 9, 2019, Accessed: May 03, 2021. [Online]. Available: https://github.com/codelucas/newspaper

[129] T. B. Brown *et al.*, “Language Models are Few-Shot Learners,” 2020, Accessed: Jun. 12, 2022. [Online]. Available: https://commoncrawl.org/the-data/

[130] S. Zhang *et al.*, “OPT: Open Pre-trained Transformer Language Models”, Accessed: Jun. 05, 2022. [Online]. Available: https://bigscience.

[131] “ChatGPT: Optimizing Language Models for Dialogue.” https://openai.com/blog/chatgpt/ (accessed Jan. 11, 2023).

[132] Y. Bengio *et al.*, “A neural probabilistic language model,” *The Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, Mar. 2003, doi: 10.5555/944919.944966.

[133] L. Fei-Fei, R. Fergus, and P. Perona, “A Bayesian approach to unsupervised one-shot learning of object categories,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2, pp. 1134–1141, 2003, doi: 10.1109/ICCV.2003.1238476.

[134] M. Fink, “Object Classification from a Single Example Utilizing Class Relevance Metrics,” in *Advances in neural information processing systems*, 2004.

[135] L. Zhang, S. Wang, and B. Liu, “Deep Learning for Sentiment Analysis: A Survey,” *Wiley Interdiscip Rev Data Min Knowl Discov*, vol. 8(4), no. e1253, 2018.

[136] F. Ricci, B. Shapira, and L. Rokach, “Recommender systems: Introduction and challenges,” in *Recommender Systems Handbook, Second Edition*, 2015. doi: 10.1007/978-1-4899-7637-6_1.

[137] E. D. Liddy, “Natural Language Processing,” 2001, Accessed: May 02, 2022. [Online]. Available: https://surface.syr.edu/istpub

[138] D. Patel, “Approaches for Sentiment Analysis on Twitter: A State-of-Art study,” 2015.

[139] D. Croce, G. Castellucci, and R. Basili, “GAN-BERT: Generative Adversarial Learning for Robust Text Classification with a Bunch of Labeled Examples,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2020, pp. 2114–2119.

[140] S. Liu, X. Cheng, F. Li, and F. Li, “TASC: Topic-adaptive sentiment classification on dynamic tweets,” *IEEE Trans Knowl Data Eng*, vol. 27, no. 6, pp. 1696–1709, Jun. 2015, doi: 10.1109/TKDE.2014.2382600.

[141] Z. Miao, Y. Li, Y. Ai, M. Labs, X. Wang, and X. Ai, “Snippext: Semi-supervised Opinion Mining with Augmented Data,” 2020, doi: 10.1145/3366423.3380144.

[142] S. Mazharul Islam, X. Dong, and G. de Melo, “Domain-Specific Sentiment Lexicons Induced from Labeled Documents,” in *Proceedings of the 28th International Conference*

on Computational Linguistics, Online, 2020, pp. 6576–6587. Accessed: Feb. 27, 2021. [Online]. Available: http://sentimentanalysis.org

[143] F. Hemmatian and M. K. Sohrabi, "A survey on classification techniques for opinion mining and sentiment analysis," *Artif Intell Rev*, vol. 52, no. 3, pp. 1495–1545, Oct. 2019, doi: 10.1007/s10462-017-9599-6.

[144] M. P. Kumar, B. Packer, and D. Koller, "Self-Paced Learning for Latent Variable Models," *NIPS*, vol. 1, p. 2, 2010.

[145] E.-P. Mastoropoulou, "Enhancing Deep Active Learning Using Selective Self-Training For Image Classification," 2019.

[146] F. Ma, D. Meng, Q. Xie, Z. Li, and X. Dong, "Self-Paced Co-training," PMLR, Jul. 2017.

[147] X. Li *et al.*, "Learning to Self-Train for Semi-Supervised Few-Shot Classification," *Adv Neural Inf Process Syst*, vol. 32, pp. 10276–10286, 2019.

[148] T. Yang, L. Hu, C. Shi, H. Ji, X. Li, and L. Nie, "HGAT: Heterogeneous Graph Attention Networks for Semi-supervised Short Text Classification," *ACM Trans Inf Syst*, vol. 39, no. 3, May 2021, doi: 10.1145/3450352.

[149] H. Kim, J. Son, and Y.-S. Han, "LST: Lexicon-Guided Self-Training for Few-Shot Text Classification," Feb. 2022, [Online]. Available: http://arxiv.org/abs/2202.02566

[150] H. Q. Abonizio, E. C. Paraiso, and S. Barbon Junior, "Toward Text Data Augmentation for Sentiment Analysis," *IEEE Transactions on Artificial Intelligence*, pp. 1–1, Sep. 2021, doi: 10.1109/tai.2021.3114390.

[151] Y. Sun *et al.*, "ERNIE: Enhanced Representation through Knowledge Integration," in *arXiv e-prints*, 2019. Accessed: May 02, 2022. [Online]. Available: https://github.com/PaddlePaddle/

[152] S. Edunov, M. Ott, M. Auli, and D. Grangier, "Understanding Back-Translation at Scale," 2018.

[153] V. Barriere and A. Balahur, "Improving Sentiment Analysis over non-English Tweets using Multilingual Transformers and Automatic Translation for Data-Augmentation," Oct. 2020, [Online]. Available: http://arxiv.org/abs/2010.03486

[154] A. Edwards, A. Ushio, J. Camacho-Collados, H. de Ribaupierre, and A. Preece, “Guiding Generative Language Models for Data Augmentation in Few-Shot Text Classification,” Nov. 2021, [Online]. Available: http://arxiv.org/abs/2111.09064

[155] K. Clark, M.-T. Luong, G. Brain, Q. V Le Google Brain, and C. D. Manning, “ELECTRA: Pre-training Text Encoders as Discriminators Rather than Generators,” in *ICLR*, 2020.

[156] D. Yarowsky, “Unsupervised word sense disambiguation rivaling supervised methods,” Association for Computational Linguistics (ACL), 1995, pp. 189–196. doi: 10.3115/981658.981684.

[157] “Self Training in Semi-supervised learning — Scikit-learn documentation.” https://scikit-learn.org/stable/modules/semi_supervised.html#self-training (accessed Nov. 15, 2021).

[158] “Stochastic gradient descent - Wikipedia.” https://en.wikipedia.org/wiki/Stochastic_gradient_descent (accessed May 26, 2021).

[159] “XGBoost Documentation — xgboost 1.5.0-SNAPSHOT documentation.” https://xgboost.readthedocs.io/en/latest/ (accessed May 16, 2021).

[160] I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization,” in *ICLR*, 2019.

[161] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, International Conference on Learning Representations, ICLR, Dec. 2015.

[162] M. Riyadh and M. O. Shafiq, “Towards Multiclass Sentiment Analysis With Limited Labeled Data,” in *IEEE International Conference on Big Data* , 2021, pp. 4955–4964.

[163] “NumPy.” https://numpy.org/ (accessed May 15, 2021).

[164] “Matplotlib: Python plotting — Matplotlib 3.4.2 documentation.” https://matplotlib.org/ (accessed May 15, 2021).

[165] “TensorFlow.” https://www.tensorflow.org/ (accessed May 15, 2021).

[166] “GitHub - crux82/ganbert: Enhancing the BERT training with Semi-supervised Generative Adversarial Networks.” https://github.com/crux82/ganbert (accessed May 26, 2021).

[167] “TensorFlow Hub - Electra Large.” https://tfhub.dev/google/electra_large/2 (accessed May 26, 2021).

[168] “Colaboratory – Google.” https://research.google.com/colaboratory/faq.html (accessed May 15, 2021).

[169] “Cloud Tensor Processing Units (TPUs) | Google Cloud.” https://cloud.google.com/tpu/docs/tpus (accessed May 02, 2022).

[170] “What Is a GPU? Graphics Processing Units Defined.” https://www.intel.com/content/www/us/en/products/docs/processors/what-is-a-gpu.html (accessed May 15, 2021).

[171] R. Socher *et al.*, “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank,” in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2013, pp. 1631–1642. Accessed: May 22, 2021. [Online]. Available: http://nlp.stanford.edu/

[172] “Twitter US Airline Sentiment | Kaggle.” https://www.kaggle.com/crowdflower/twitter-airline-sentiment (accessed May 22, 2021).

[173] S. Rosenthal, N. Farra, and P. Nakov, “SemEval-2017 Task 4: Sentiment Analysis in Twitter,” in *Proceedings of the 11th International Workshop on Semantic Evaluation ({S}em{E}val-2017)*, Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 502–518. doi: 10.18653/v1/S17-2088.

[174] P. G. de Vries, “Stratified Random Sampling,” in *Sampling Theory for Forest Inventory*, Springer, 1986, pp. 31–55.

[175] J. Dai, H. Yan, T. Sun, P. Liu, and X. Qiu, “Does syntax matter? A strong baseline for Aspect-based Sentiment Analysis with RoBERTa,” in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* , 2021, pp. 1816–1829. Accessed: May 05, 2023. [Online]. Available: https://github.com/

[176] “sklearn.metrics.f1_score — scikit-learn 1.2.2 documentation.” https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score (accessed May 05, 2023).

[177] T. Mitchell, *Machine Learning*. McGraw-Hill International, 1997.

[178] J. Brownlee, “Statistical Methods for Machine Learning Discover how to Transform Data into Knowledge with Python,” 2019.

[179] M. Grandini, E. Bagli, G. Visani, M. Grandini, E. Bagli, and G. Visani, “Metrics for Multi-Class Classification: an Overview,” *ArXiv*, p. arXiv:2008.05756, Aug. 2020, doi: 10.48550/ARXIV.2008.05756.

[180] F. Wilcoxon, “Individual comparisons by ranking methods, Biometrics Bulletin, 1, 80–83.(1947) Probability tables for individual comparisons by ranking methods.” Biomet, 1945.

[181] J. Derrac, S. García, D. Molina, and F. Herrera, “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms,” *Swarm Evol Comput*, vol. 1, pp. 3–18, 2011, doi: 10.1016/j.swevo.2011.02.002.

[182] J. Demšar, “Statistical Comparisons of Classifiers over Multiple Data Sets,” *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[183] Y. Kim and O. Zhang, “Credibility Adjusted Term Frequency: A Supervised Term Weighting Scheme for Sentiment Analysis and Text Classification,” pp. 79–83, 2014, Accessed: May 05, 2023. [Online]. Available: https://www.cs.cornell.edu/people/pabo/movie-review-

[184] Y. Guo, Q. Hu, M. Cordy, M. Papadakis, and Y. Le Traon, “DRE: density-based data selection with entropy for adversarial-robust deep learning models,” *Neural Comput Appl*, vol. 35, no. 5, pp. 4009–4026, Feb. 2023, doi: 10.1007/S00521-022-07812-2/TABLES/6.

[185] S. Zhang, X. Zhang, and J. Chan, “A word-character convolutional neural network for language-agnostic twitter sentiment analysis,” *ACM International Conference Proceeding Series*, vol. 2017-December, Dec. 2017, doi: 10.1145/3166072.3166082.

[186] B. Ko and H. J. Choi, “Twice fine-tuning deep neural networks for paraphrase identification,” *Electron Lett*, vol. 56, no. 9, pp. 449–450, Apr. 2020, doi: 10.1049/el.2019.4183.

[187] "Optimization: Stochastic Gradient Descent." http://deeplearning.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/ (accessed May 02, 2022).

[188] A. Zadeh, M. Chen, S. Poria, E. Cambria, and L.-P. Morency, "Tensor Fusion Network for Multimodal Sentiment Analysis," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017, pp. 1103–1114.

[189] Z. Li, Y. Zou, C. Zhang, Q. Zhang, and Z. Wei, "Learning Implicit Sentiment in Aspect-based Sentiment Analysis with Supervised Contrastive Pre-Training," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021, pp. 246–256. Accessed: May 05, 2023. [Online]. Available: https://github.com/Tribleave/SCAPT-ABSA

[190] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," 2017.

[191] M. Riyadh and M. O. Shafiq, "GAN-BElectra: Enhanced Multi-class Sentiment Analysis with Limited Labeled Data," *https://doi.org/10.1080/08839514.2022.2083794*, vol. 36, no. 1, 2022, doi: 10.1080/08839514.2022.2083794.

[192] M. Riyadh and M. Omair Shafiq, "Towards Multi-class Sentiment Analysis with Limited Labeled Data," *Proceedings - 2021 IEEE International Conference on Big Data, Big Data 2021*, pp. 4955–4964, 2021, doi: 10.1109/BIGDATA52589.2021.9671692.

[193] M. Hardalov, A. Arora, P. Nakov, and I. Augenstein, "Few-Shot Cross-Lingual Stance Detection with Sentiment-Based Pre-Training," Sep. 2021, [Online]. Available: http://arxiv.org/abs/2109.06050

[194] M. Li, H. Zhao, H. Su, Y. R. Qian, and P. Li, "Emotion-cause span extraction: a new task to emotion cause identification in texts," *Applied Intelligence*, vol. 51, no. 10, pp. 7109–7121, Oct. 2021, doi: 10.1007/s10489-021-02188-7.

[195] F. Wang, Z. Ding, R. Xia, and J. Li Zhaoyu and Yu, "Multimodal Emotion-Cause Pair Extraction in Conversations," Oct. 2021.

[196] S. Dutta, D. Das, and T. Chakraborty, "Changing views: Persuasion modeling and argument extraction from online discussions," *Inf Process Manag*, vol. 57, no. 2, p. 102085, Mar. 2020, doi: 10.1016/J.IPM.2019.102085.

[197] S. Yu, G. D. S. Martino, and P. Nakov, “Experiments in Detecting Persuasion Techniques in the News,” Nov. 2019, [Online]. Available: http://arxiv.org/abs/1911.06815

[198] H. Chen, D. Ghosal, N. Majumder, A. Hussain, and S. Poria, “Persuasive Dialogue Understanding: the Baselines and Negative Results,” Nov. 2020, [Online]. Available: http://arxiv.org/abs/2011.09954

[199] S. Das and A. K. Kolya, “Predicting the pandemic: sentiment evaluation and predictive analysis from large-scale tweets on Covid-19 by deep convolutional neural network,” *Evol Intell*, vol. 15, no. 3, pp. 1913–1934, Sep. 2022, doi: 10.1007/S12065-021-00598-7/FIGURES/7.

[200] A. Ramesh *et al.*, “Zero-Shot Text-to-Image Generation,” Feb. 2021, doi: 10.48550/arxiv.2102.12092.

[201] S. Yada, K. Ikeda, and K. Hoashi Keiichiro and Kageura, “A bootstrap method for automatic rule acquisition on emotion cause extraction,” in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, IEEE, Nov. 2017.

[202] C. Donahue, M. Lee, and P. Liang, “Enabling Language Models to Fill in the Blanks,” May 2020, [Online]. Available: http://arxiv.org/abs/2005.05339

[203] S. Y. M. Lee, Y. Chen, C. R. Huang, and S. Li, “Detecting Emotion Causes With A Linguistic Rule-based Approach,” *Comput Intell*, vol. 29, no. 3, pp. 390–416, Aug. 2013, doi: 10.1111/J.1467-8640.2012.00459.X.

[204] Y. Chen, W. Hou, and X. Cheng, “Hierarchical convolution neural network for emotion cause detection on microblogs,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11139 LNCS, pp. 115–122, 2018, doi: 10.1007/978-3-030-01418-6_12/COVER/.

[205] B. Xu, H. Lin, Y. Lin, Y. Diao, L. Yang, and K. Xu, “Extracting Emotion Causes Using Learning to Rank Methods from an Information Retrieval Perspective,” *IEEE Access*, vol. 7, pp. 15573–15583, 2019, doi: 10.1109/ACCESS.2019.2894701.

[206] R. Xia and Z. Ding, “Emotion-Cause Pair Extraction: A New Task to Emotion Analysis in Texts.”

[207] C. Fan, C. Yuan, J. Du, L. Gui, M. Yang, and R. Xu, “Transition-based Directed Graph Construction for Emotion-Cause Pair Extraction,” pp. 3707–3717, Jul. 2020, doi: 10.18653/V1/2020.ACL-MAIN.342.

[208] S. Smith *et al.*, “Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model”.

[209] F. Petroni *et al.*, “Language Models as Knowledge Bases?”, Accessed: Jun. 12, 2022. [Online]. Available: https://github.com/pytorch/fairseq

[210] R. Zellers *et al.*, “Defending Against Neural Fake News”, Accessed: Jun. 12, 2022. [Online]. Available: https://rowanzellers.com/grover

[211] A. See, A. Pappu, R. Saxena, A. Yerukola, and C. D. Manning, “Do Massively Pretrained Language Models Make Better Storytellers?”, Accessed: Jun. 12, 2022. [Online]. Available: https://github.com/huggingface/

[212] M. Joshi *et al.*, “SpanBERT: Improving Pre-training by Representing and Predicting Spans”, Accessed: Feb. 19, 2022. [Online]. Available: https://github.com/facebookresearch/

[213] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: a Method for Automatic Evaluation of Machine Translation”.

[214] S. Banerjee and A. Lavie, “METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments”.

[215] C.-Y. Lin, “ROUGE: A Package for Automatic Evaluation of Summaries”.

[216] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “BERTSCORE: EVALUATING TEXT GENERATION WITH BERT”, Accessed: Jun. 12, 2022. [Online]. Available: https://github.com/Tiiiger/bert_score.

[217] R. Flesch, “A new readability yardstick,” *Journal of Applied Psychology*, vol. 32, no. 3, pp. 221–233, Jun. 1948, doi: 10.1037/H0057532.

[218] “Coleman–Liau index - Wikipedia.” https://en.wikipedia.org/wiki/Coleman%E2%80%93Liau_index (accessed Jun. 12, 2022).

[219] W. Zhu and S. Bhat, “GRUEN for Evaluating Linguistic Quality of Generated Text”, Accessed: May 16, 2022. [Online]. Available: http://tac.nist.gov/

[220] “cardiffnlp/twitter-roberta-base-sentiment · Hugging Face.” https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment (accessed Jan. 08, 2023).

[221] “j-hartmann/emotion-english-distilroberta-base · Hugging Face.” https://huggingface.co/j-hartmann/emotion-english-distilroberta-base (accessed May 06, 2023).

[222] J. Novikova, O. Dušek, A. C. Curry, and V. Rieser, “Why We Need New Evaluation Metrics for NLG”, Accessed: Jun. 12, 2022. [Online]. Available: https://github.com/glampouras/JLOLS_

[223] “language-check · PyPI.” https://pypi.org/project/language-check/ (accessed Jun. 12, 2022).

[224] D. Ping, “The Machine Learning Solutions Architect Handbook,” p. 442, 2022, Accessed: Jan. 08, 2023. [Online]. Available: https://www.oreilly.com/library/view/the-machine-learning/9781801072168/

[225] G. Bonaccorso, “Machine Learning Algorithms: Popular algorithms for data science and machine ... - Giuseppe Bonaccorso - Google Libros,” *Packt Publishing*, vol. 2nd ed, pp. 0–332, 2018, Accessed: Jan. 08, 2023. [Online]. Available: https://www.oreilly.com/library/view/machine-learning-algorithms/9781789347999/

[226] M. Riyadh and M. O. Shafiq, “GAN-BElectra: Enhanced Multi-class Sentiment Analysis with Limited Labeled Data,” *https://doi.org/10.1080/08839514.2022.2083794*, vol. 36, no. 1, 2022, doi: 10.1080/08839514.2022.2083794.

[227] M. Riyadh and M. Omair Shafiq, “Towards Multi-class Sentiment Analysis with Limited Labeled Data,” *Proceedings - 2021 IEEE International Conference on Big Data, Big Data 2021*, pp. 4955–4964, 2021, doi: 10.1109/BIGDATA52589.2021.9671692.

[228] J. Zhou, J. Tian, R. Wang, Y. Wu, W. Xiao, and L. He, “SentiX: A Sentiment-Aware Pre-Trained Model for Cross-Domain Sentiment Analysis,” pp. 568–579, Jan. 2020, doi: 10.18653/V1/2020.COLING-MAIN.49.

[229] Y. Liu and M. Lapata, “Text Summarization with Pretrained Encoders,” *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th*

International Joint Conference on Natural Language Processing, Proceedings of the Conference, pp. 3730–3740, Aug. 2019, doi: 10.48550/arxiv.1908.08345.

[230] Y. Niu, F. Huang, J. Liang, W. Chen, X. Zhu, and M. Huang, "A Semantic-based Method for Unsupervised Commonsense Question Answering," *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pp. 3037–3049, May 2021, doi: 10.48550/arxiv.2105.14781.

[231] L. Cui *et al.*, "Template-Based Named Entity Recognition Using BART," *ACL-IJCNLP*, 2021.

[232] Q. Gao *et al.*, "Overview of NTCIR-13 ECA Task."

[233] A. Singh, S. Hingane, S. Wani, and A. Modi, "An End-to-End Network for Emotion-Cause Pair Extraction," Mar. 2021, [Online]. Available: http://arxiv.org/abs/2103.01544

[234] W. Fan, Y. Zhu, Z. Wei, T. Yang, W. H. Ip, and Y. Zhang, "Order-guided deep neural network for emotion-cause pair prediction," *Appl Soft Comput*, vol. 112, Nov. 2021, doi: 10.1016/j.asoc.2021.107818.

[235] W. Zhu, Z. Hu, and E. P. Xing, "Text Infilling," 2019, Accessed: Feb. 12, 2022. [Online]. Available: https://github.com/VegB/Text_Infilling

[236] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-Shot Learning-A Comprehensive Evaluation of the Good, the Bad and the Ugly".

[237] D. Pascual, B. Egressy, C. Meister, R. Cotterell, and R. Wattenhofer, "A Plug-and-Play Method for Controlled Text Generation," Sep. 2021, [Online]. Available: http://arxiv.org/abs/2109.09707

[238] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward Controlled Generation of Text," 2017.

[239] S. Kumar, E. Malmi, A. Severyn, and Y. Tsvetkov, "Controlled Text Generation as Continuous Optimization with Multiple Constraints", Accessed: Jun. 25, 2022. [Online]. Available: https://github.com/Sachin19/mucoco

[240] I. Singh, A. Barkati, T. Goswamy, and A. Modi, “Adapting a Language Model for Controlled Affective Text Generation,” Nov. 2020, [Online]. Available: http://arxiv.org/abs/2011.04000

[241] N. Madaan, I. Padhi, N. Panwar, and D. Saha, “Generate Your Counterfactuals: Towards Controlled Counterfactual Generation for Text,” 2021. [Online]. Available: www.aaai.org

[242] K. Yang and D. Klein, “FUDGE: Controlled Text Generation With Future Discriminators”, Accessed: Jun. 25, 2022. [Online]. Available: https://github.com/yangkevin2/

[243] K. Yang *et al.*, “Tailor: A Prompt-Based Approach to Attribute-Based Controlled Text Generation.”

[244] B. Lester, R. Al-Rfou, and N. Constant, “The Power of Scale for Parameter-Efficient Prompt Tuning,” *EMNLP 2021 - 2021 Conference on Empirical Methods in Natural Language Processing, Proceedings*, pp. 3045–3059, Apr. 2021, doi: 10.48550/arxiv.2104.08691.

[245] M. Ghazvininejad, V. Karpukhin, V. Gor, and A. Celikyilmaz, “Discourse-Aware Soft Prompting for Text Generation.”

[246] S. Wu, F. Chen, F. Wu, Y. Huang, and X. Li, “A Multi-Task Learning Neural Network for Emotion-Cause Pair Extraction.” Accessed: Feb. 21, 2021. [Online]. Available: https://github.com/wusx00/MTNECP

[247] “Natural language processing - Research - Cardiff University.” https://www.cardiff.ac.uk/research/explore/research-units/natural-language-processing (accessed May 06, 2023).

[248] “vennify/t5-base-grammar-correction · Hugging Face.” https://huggingface.co/vennify/t5-base-grammar-correction (accessed Jan. 08, 2023).

[249] “facebook/opt-1.3b · Hugging Face.” https://huggingface.co/facebook/opt-1.3b (accessed Jan. 08, 2023).

[250] “Google Colab.” https://research.google.com/colaboratory/faq.html (accessed Jan. 08, 2023).

[251] “gpt2-xl · Hugging Face.” https://huggingface.co/gpt2-xl (accessed Jan. 08, 2023).

[252] "OPTConfig." https://huggingface.co/docs/transformers/model_doc/opt#transformers.OPTConfig (accessed Jan. 11, 2023).

[253] "GPT2 Config." https://huggingface.co/docs/transformers/model_doc/gpt2#transformers.GPT2Config (accessed Jan. 11, 2023).

[254] "Generation." https://huggingface.co/docs/transformers/v4.24.0/en/main_classes/text_generation#transformers.GenerationConfig (accessed Jan. 11, 2023).

[255] Z. Yang, S. Ram, and F. Currim, "What Makes an Online Suggestion A Good One for Online Health Communities Communities," 2000, Accessed: Nov. 10, 2022. [Online]. Available: https://aisel.aisnet.org/amcis2020/healthcare_it/healthcare_it/1

[256] L. Hanu, "Detoxify," *Github*, 2020. https://github.com/unitaryai/detoxify (accessed Jan. 08, 2023).

[257] F. Barbieri, J. Camacho-Collados, L. Neves, and L. Espinosa-Anke, "TWEETEVAL: Unified Benchmark and Comparative Evaluation for Tweet Classification", Accessed: Jan. 08, 2023. [Online]. Available: https://github.com/cardiffnlp/tweeteval

[258] HF Canonical Model Maintainers, "distilbert-base-uncased-finetuned-sst-2-english," *HuggingFace*, 2022. https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english (accessed Jan. 08, 2023).

[259] M. Moshkov, "Greedy Algorithms," *Wiley Encyclopedia of Electrical and Electronics Engineering*, pp. 1–11, Jun. 2015, doi: 10.1002/047134608X.W8262.

[260] "rominf/profanity-filter: A Python library for detecting and filtering profanity." https://github.com/rominf/profanity-filter (accessed Jan. 08, 2023).

[261] "Qualtrics XM: The Leading Experience Management Software." https://www.qualtrics.com/ (accessed Jan. 08, 2023).

[262] "Amazon Mechanical Turk." https://www.mturk.com/ (accessed Jan. 08, 2023).

[263] “How to Label Response Scale Points in Your Survey | Qualtrics.” https://www.qualtrics.com/blog/how-to-label-response-scale-points-in-your-survey-to-avoid-misdirecting-respondents/ (accessed Jan. 08, 2023).

[264] K. S. Dobson A N D and K. J. Mothersill, “Equidistant Categorical Labels For Construction Of Likert-type Scales,” 1979.

[265] “About Python™ | Python.org.” https://www.python.org/about/ (accessed Jan. 08, 2023).

[266] “language-tool-python · PyPI.” https://pypi.org/project/language-tool-python/ (accessed Jan. 08, 2023).

[267] “huggingface (Hugging Face).” https://huggingface.co/huggingface (accessed Jan. 08, 2023).

[268] “SPSS Software | IBM.” https://www.ibm.com/spss (accessed Jan. 08, 2023).

[269] “Microsoft Excel Spreadsheet Software | Microsoft 365.” https://www.microsoft.com/en-us/microsoft-365/excel (accessed Jan. 08, 2023).

[270] “pandas - Python Data Analysis Library.” https://pandas.pydata.org/ (accessed Jan. 08, 2023).

[271] L. Hansen and K. Enevoldsen, “TextDescriptives: A Python package for calculating a large variety of statistics from text,” Jan. 2023, doi: 10.48550/arxiv.2301.02057.

[272] “Explore / Twitter.” https://twitter.com/ (accessed Jan. 08, 2023).

[273] “Snscrape: A social networking service scraper in Python.” https://github.com/JustAnotherArchivist/snscrape (accessed Jan. 08, 2023).

[274] “Measure reading levels with readability indexes.” https://www.wyliecomm.com/2021/11/measure-reading-levels-with-readability-indexes/ (accessed Jan. 08, 2023).

[275] “Readability formulas – Readable.” https://readable.com/features/readability-formulas/ (accessed Jan. 08, 2023).

[276] “The Chi-Square Test | Introduction to Statistics | JMP.” https://www.jmp.com/en_be/statistics-knowledge-portal/chi-square-test.html (accessed Jan. 08, 2023).

[277] “NLP vs. NLU vs. NLG: the differences between three natural language processing concepts - Watson Blog.” https://www.ibm.com/blogs/watson/2020/11/nlp-vs-nlu-vs-nlg-the-differences-between-three-natural-language-processing-concepts/ (accessed Jan. 20, 2023).

[278] W. Zhao, M. Peyrard, F. Liu, Y. Gao, C. M. Meyer, and S. Eger, “MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance”, Accessed: Feb. 18, 2023. [Online]. Available: http://tiny.cc/vsqtbz

[279] L. Ouyang *et al.*, “Training language models to follow instructions with human feedback,” Mar. 2022, doi: 10.48550/arxiv.2203.02155.

[280] “Model index for researchers - OpenAI API.” https://platform.openai.com/docs/model-index-for-researchers (accessed Feb. 18, 2023).

[281] B. Guo *et al.*, “How Close is ChatGPT to Human Experts? Comparison Corpus, Evaluation, and Detection”, Accessed: Feb. 11, 2023. [Online]. Available: https://chat.openai.com/chat

[282] C. Mellish and R. Dale, “Evaluation in the context of natural language generation,” *Comput Speech Lang*, vol. 12, no. 4, pp. 349–373, Oct. 1998, doi: 10.1006/CSLA.1998.0106.

[283] J. Coch, “Evaluating and comparing three text-production techniques,” in *COLING*, 1996.

[284] S. Bangalore, O. Rambow, and S. Whittaker, “Evaluation Metrics for Generation”.

[285] T. Marciniak and M. Strube, “Classification-based generation using TAG,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3123, pp. 100–109, 2004, doi: 10.1007/978-3-540-27823-8_11/COVER.

[286] I. Langkilde-Geary, “An Empirical Verification of Coverage and Correctness for a General-Purpose Sentence Generator”.

[287] E. Reiter and S. Sripada, “Should Corpora Texts Be Gold Standards for NLG?” pp. 97–104, 2002. Accessed: Feb. 18, 2023. [Online]. Available: https://aclanthology.org/W02-2113

[288] A. Belz and E. Reiter, “Comparing Automatic and Human Evaluation of NLG Systems”.

[289] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “BERTSCORE: EVALUATING TEXT GENERATION WITH BERT”, Accessed: May 16, 2022. [Online]. Available: https://github.com/Tiiiger/bert

[290] P. Colombo, G. Staerman, C. Clavel, and P. Piantanida, “Automatic Text Evaluation through the Lens of Wasserstein Barycenters,” *EMNLP 2021 - 2021 Conference on Empirical Methods in Natural Language Processing, Proceedings*, pp. 10450–10466, Aug. 2021, doi: 10.48550/arxiv.2108.12463.

[291] W. Zhao, Y. Gao, and S. Eger, “Evaluating Machine Translation without Human References Using Cross-lingual Encoders”.

[292] Y. Song, J. Zhao, and L. Specia, “SentSim: Crosslingual Semantic Evaluation of Machine Translation,” *NAACL-HLT 2021 - 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*, pp. 3143–3156, 2021, doi: 10.18653/V1/2021.NAACL-MAIN.252.

[293] C. Leiter, P. Lertvittayakumjorn, M. Fomicheva, W. Zhao, Y. Gao, and S. Eger, “Towards Explainable Evaluation Metrics for Natural Language Generation”, Accessed: Feb. 18, 2023. [Online]. Available: https://github.com/Gringham/explainable-metrics-machine-translation.

[294] B. Marie, A. Fujita, and R. Rubino, “Scientific Credibility of Machine Translation Research: A Meta-Evaluation of 769 Papers,” *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pp. 7297–7306, Jun. 2021, doi: 10.48550/arxiv.2106.15195.

[295] J. Grünwald, C. Leiter, and S. Eger, “Can we do that simpler? Simple, Efficient, High-Quality Evaluation Metrics for NLG”.

[296] R. Sharma, J. F. Allen, O. Bakhshandeh, and N. Mostafazadeh, “Tackling the Story Ending Biases in The Story Cloze Test,” *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, vol. 2, pp. 752–757, 2018, doi: 10.18653/V1/P18-2119.

[297] “Google Sheets: Online Spreadsheet Editor | Google Workspace.” https://www.google.com/sheets/about/ (accessed Feb. 18, 2023).

[298] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, “GLUE: A MULTI-TASK BENCHMARK AND ANALYSIS PLATFORM FOR NATURAL LANGUAGE UNDERSTAND-ING”.

[299] Y. Yang, W.-T. Yih, and C. Meek, “WIKIQA: A Challenge Dataset for Open-Domain Question Answering,” pp. 17–21, 2015, Accessed: Feb. 18, 2023. [Online]. Available: http://aka.ms/WikiQA.

[300] “Google AI updates: Bard and new AI features in Search.” https://blog.google/technology/ai/bard-google-ai-search-updates/ (accessed Feb. 18, 2023).

[301] “Localization vs. Internationalization,” *W3C*, Accessed: May 06, 2023. [Online]. Available: http://www.w3.org/International/questions/qa-i18n