Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Martin Johannes Nilsen
Ole Jonas Liahagen

From Words to Weapons

Uncovering Potential School Shooters through Linguistic Cues in Written Posts

Master's thesis in Computer Science
Supervisor: Björn Gambäck
June 2023

NTNU
Norwegian University of
Science and Technology

Martin Johannes Nilsen
Ole Jonas Liahagen

From Words to Weapons

Uncovering Potential School Shooters through Linguistic Cues in Written Posts

Master's thesis in Computer Science
Supervisor: Björn Gambäck
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

With the reported amount of daily users of social media and online forums being in the billions, these online platforms are host to all kinds of people. Allowing for free speech and relative anonymity, the users have, in some cases, the liberty to express themselves in any way they desire without being moderated or censored. Investigations after school shooting incidents have shown that a large number of attackers leave behind written texts in the form of online posts or handwritten documents. In most cases, these texts are produced before and leading up to the attack. Furthermore, previous work on text classification and sentiment analysis has shown that significant information about a person can be retrieved from their writings. This raises the possibility that extracting indicators from the texts of previous school shooters could aid in identifying warning signs of a potential future school shooting before it takes place.

For the purposes of this study, a collection of 3028 texts written by 26 distinct school shooters leading up to their attacks has been collected and annotated. Analysis of this dataset reveals similar psycholinguistic and statistical traits between the texts of school shooters, separating them from texts written by non-shooters. This analysis, in addition to related work found in a preliminary literature review, led to the application of Linguistic Inquiry and Word Count (LIWC) and Term Frequency-Inverse Document Frequency (TF-IDF) features for classification, albeit yielding limited results. A further investigation into the more subtle linguistic cues of a school shooter was performed by utilizing globally pretrained word embeddings, with Global Vectors for Word Representation (GloVe), Fast Text Encoding using a pre-trained Character-level Model (FastText), and Bidirectional Encoder Representations from Transformers (BERT) as feature inputs to both classical and deep learning models. A host of large language models were additionally employed to test their predictive power on the school shooter dataset. Ultimately, a voting classifier based on the best performing models from each experiment was constructed. Although tested on a small number of perpetrators, the results of the application of our final voting based classifier are promising, beating previous similar studies' performance when screening for school shooter texts and achieving an F_2-score of 0.9656.

Sammendrag

Sosiale medier og nettbaserte forum anslås å bli brukt av milliarder hver dag, som plattformer for åpen dialog og i noen tilfeller helt umoderert diskusjon. I tillegg, viser utredninger i etterkant av skoleskytinger at mange skoleskytere etterlater seg tekst i form av enten nettbaserte poster eller håndskrevne dokumenter. I de fleste tilfeller er disse tekstene skrevet i forkant av et angrep. Samtidig, viser forskning på tekstklassifisering og sentimentanalyse at informasjon om forfatteren bak en tekst kan bli uthentet, enten i form av gjenkjennelige tegn eller mer subtile hint. Ved å forsøke å identifisere varseltegn fra tekster skrevet av skoleskytere i forkant av et tidligere angrep, kan man potensielt ta i bruk denne kunnskapen for å forhindre nye tilfeller før de finner sted.

En samling av 3028 unike tekster skrevet av 26 tidligere skoleskytere har blitt samlet inn i denne masterstudien. Analyser av tekstene viser at det kan finnes psykolingvistiske og statistiske trekk som skiller tekster skrevet av skoleskytere fra tekster skrevet av ikke-skoleskytere. Med utgangspunkt i tidligere studier og denne innledende granskningen, ble karakteristiske trekk ved tekstene uthentet ved hjelp av rammeverkene Linguistic Inquiry and Word Count (LIWC) og Term Frequency-Inverse Document Frequency (TF-IDF), med begrenset grad av suksess. For å trekke ut potensiell informasjon skjult i mer subtile forskjeller i tekster skrevet av skoleskytere kontra ikke-skoleskytere, ble det tatt i bruk opptrente ordvektorer (word embeddings) som data til trening av maskinlæringsmodeller. Både klassiske modeller og nevrale nettverk ble trent og testet på ordvektorer generert av modellene Global Vectors for Word Representation (GloVe), Fast Text Encoding using a pre-trained Character-level Model (FastText) og Bidirectional Encoder Representations from Transformers (BERT). Et utvalg språkmodeller ble så anvendt for å teste deres treffsikkerhet på klassifiseringsproblemet. Avslutningsvis ble de beste modellene fra hvert eksperiment trukket ut og samlet i en ensemble-modell hvor de underliggende modellene stemmer på klassetilhørligheten til tekstene de blir presentert. Til tross for et forholdsvis lite utvalg tekster, på grunn av tilgjengeligheten på slik data, er resultatene fra vår ensemble-løsning lovende. Den endelige modellen overgår tidligere studier på deteksjon av tekster skrevet av skoleskytere, og oppnår en endelig F_2-score på 0.9656.

Preface

This master's thesis represents the final step towards obtaining a Master of Science degree in Computer Science from the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. It expands on a specialization project undertaken in the Fall of 2022, with research conducted from January to June 2023. The thesis was supervised by Björn Gambäck and carried out within the Data and Artificial Intelligence Group at the Department of Computer Science.

The motivation for this thesis arose from a desire to explore the fields of Natural Language Processing (NLP) and Machine Learning (ML), specifically within the topics of hate speech and threat detection. Following a preliminary literature review performed in the specialization project, we wanted to delve deeper into methodologies used for detecting school shooters based on textual information. Our attention was centered on exploring the application of new and combined feature sets while also leveraging the latest developments in state-of-the-art machine learning models. The objective was to develop a majority vote solution, combining effective methods from existing research with other techniques, to improve the detection of potential school shooters through linguistic cues.

The reader is expected to have a fundamental knowledge of machine learning and the evaluation metrics used for supervised learning tasks. Additionally, familiarity with natural language processing will be beneficial, particularly in how text is processed, represented, and features are extracted for computational processing. However, the relevant theory will be elaborated upon in detail to ensure a solid understanding of the methodologies used in this thesis. Finally, a basic knowledge of linear algebra, calculus, probability, and statistics could be advantageous to fully understand the applied methods; however, it is possible to understand the theory without fully understanding the underlying mathematics.

Acknowledgments

We would like to extend a special thanks to the people valuable for the outcome of this thesis. First, thank you, Björn Gambäck, for your invaluable guidance, insightful feedback, and engaging conversations throughout this process. Second, we would like to express our sincere gratitude to Peter Langman for his contribution to this thesis by kindly allowing us to use his extensive database on school shooters and their written work. Finally, we are forever thankful for the wonderful individuals we have met during our academic journey; our five years at NTNU would not have been the same without them.

> Jonas, it's been an incredible journey, hasn't it? From the bachelor's to the master's, every group project, every thesis, all tackled together. It wasn't just about the work, but also the unforgettable memories and a whole lot of fun. I can't thank you enough for not just being an amazing study partner but also a great friend. As we close this chapter, I'm excited to embark on a new one, knowing we've formed a bond that will last a lifetime. Here's to many more years of shared success, friendship, and memories yet to come. Cheers to us!
>
> — Martin

> Thank you for 5 good years here at NTNU Martin. It has been a pleasure working and studying with you through all these years. As we deliver this final project, I thank you for all this time spent together and look forward to what is to come. Judging from our time together I know you'll do great in your future endeavors. I'm excited to see what the future hold in store for you!
>
> — Jonas

Martin Johannes Nilsen & Ole Jonas Liahagen
Trondheim, 10th June 2023

Contents

List of Figures

List of Tables

Abbreviations

AI Artificial Intelligence

ALBERT A Lite BERT

ANN Artificial Neural Network

API Application Programming Interface

BERT Bidirectional Encoder Representations from Transformers

biLSTM Bidirectional Long Short-Term Memory

BOW Bag of Words

CBOW Continous Bag of Words

CHAID Chi-squared Automatic Interaction Detection

CNN Convolutional Neural Network

DF Document Frequency

DistilBERT Distilled Version of BERT

EmoLex Emotion Lexicon

ERG22+ Extremism Risk Guidelines

FastText Fast Text Encoding using a pre-trained Character-level Model

FFM Five-Factor Model

FN False Negative

FNN Feedforward Neural Network

FP False Positive

GloVe Global Vectors for Word Representation

GNB Gaussian Naïve Bayes

GP Gaussian Processes

GRU Gated Recurrent Unit

IDF Inverse Document Frequency

KNN K-Nearest Neighbors

LASSO Least Absolute Shrinkage and Selection Operator

LDA Linear Discriminant Analysis

LIWC Linguistic Inquiry and Word Count

LLM Large Language Model

LSTM Long Short-Term Memory

MAE Mean Absolute Error

MBTI Myers-Briggs Type Indicator

ML Machine Learning

MLG Multi-Level Guidelines

MLM Masked Language Modeling

MLP Multi-Layer Perceptron

MNB Multinomial Naïve Bayes

MSE Mean Square Error

NB Naïve Bayes

NLP Natural Language Processing

NLTK Natural Language Toolkit

NSP Next Sentence Prediction

NTNU Norwegian University of Science and Technology

OMD Obama-McCain Debate

PCA Principal Component Analysis

PDF Portable Document Format

POS Part-of-Speech

RBF Radial Basis Function

RegEx Regular Expression

RF Random Forests

RMSE Root Mean Square Error

RNN Recurrent Neural Network

RoBERTa Robustly Optimized BERT Pretraining Approach

SLP Single-Layer Perceptron

SLR Structured Literature Review

SMO Simple Minimal Optimization

SMOTE Synthetic Minority Over-sampling Technique

SNA Social Network Analysis

SPLICE Structured Programming for Linguistic Cue Extraction

SVM Support Vector Machine

TF Term Frequency

TF-IDF Term Frequency-Inverse Document Frequency

TN True Negative

TP True Positive

TRAP-18 Terrorist Radicalization Assessment Protocol

VERA-2R Violent Extremism Risk Assessment 2 Revised

Word2Vec Word to Vector

XGBoost Extreme Gradient Boosting

Glossary

Artificial intelligence A field of study focused on computer systems' ability to simulate human intelligence processes

Artificial neural network A computational model that is inspired by the structure and function of the human brain, designed to recognize patterns in data through a network of interconnected nodes

Batch size A hyperparameter that determines the number of examples to send into the model for each iteration in an epoch, and is collectively part of forming the error, which is a measurement of needed adjustment

Big Five A widely used framework in personality psychology, also known as the Five-Factor Model (FFM), that classifies individuals based on five fundamental personality traits: Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism

Computer science The study of computational systems and computers, which encompasses the theoretical understanding, design, development, and application of software and hardware

Convolutional neural network A neural network variant optimized for handling grid-like data, including images or structured texts, employing convolutional layers for feature detection and dimensionality reduction

Epoch One whole iteration over the entire training set

Feedforward neural network An artificial neural network where data flows unidirectionally from input to output, ideal for classification or regression tasks

Information retrieval The process of searching for and retrieving recorded data and information from a file or database

Learning rate A hyperparameter determining the step size during the machine learning optimization process, influencing the speed of convergence towards the loss function's minimum

Machine learning A subfield of artificial intelligence, devoted to understanding and building methods that leverage data to improve performance, thus "learning" based on given data

Myers-Briggs Type Indicator A psychological tool categorizing individuals into one of 16 personality types based on four key dimensions: Extraversion/Introversion, Sensing/Intuition, Thinking/Feeling, and Judging/Perceiving

Natural language processing A subfield of artificial intelligence, enabling computers to understand and interpret human language

One-hot encoding A technique for transforming categorical variables into binary vectors, with each dimension representing a unique category. In natural language processing, it is used to convert words or phrases into these binary vectors

Pearson correlation coefficient A measure of linear correlation between two sets of data, in the form of a normalized value between -1 and 1

Principal component analysis A statistical technique that reduces the dimensionality of a dataset by transforming it into a new set of orthogonal variables called principal components, which capture the most variance in the data

Recurrent neural network A type of neural network that allows information to flow in cycles, enabling them to process sequential data and handle temporal dependencies

Regularization A strategy used in machine learning to prevent overfitting by imposing constraints on the model parameters

Supervised learning A subcategory of machine learning in which models are trained on labeled data, learning to predict outcomes by mapping input data to the corresponding label, also known as the "target"

Unsupervised learning A subcategory of machine learning that uses unlabeled data to uncover hidden patterns and structures, enabling models to learn from the intrinsic properties of the input without predetermined targets

Weight decay Also known as L2 regularization, this technique mitigates overfitting in neural networks by adding a penalty term to the loss function, which encourages smaller weight magnitudes

1. Introduction

School shootings have become an increasingly alarming concern worldwide. The loss of innocent lives to such heinous acts of violence is a painful reminder of the urgent need to address this problem. With the rise of social media and online platforms, many potential perpetrators express their intentions, ideologies, or emotional distress through their online posts prior to committing an attack. Recognizing the potential of this digital footprint to serve as an early warning system, this thesis focuses on uncovering potential school shooters by analyzing their written posts published online. The goal is to delve deeper into methodologies for identifying threat cues, with the objective of facilitating timely intervention before a possible school shooting takes place.

This introductory chapter lays a foundation for the thesis, starting with exploring the background and motivation behind the study. Following this theoretical grounding, the next section aims to define the research goal and poses the specific research questions to be tackled. Then, the research method will be presented, covering the strategies and techniques that will be employed, before the contributions of this study will be listed. Finally, the forthcoming chapters will be outlined, offering a glimpse into the structure and progression of the thesis.

1.1 Background and Motivation

Over the past five decades, there has been a significant surge in attacks committed by lone wolf perpetrators — individuals unaffiliated with any organized group. The Global Terrorism Index reports that the proportion of such attackers has had a rapid increase from 5% in 1970 to over 70% between 2014 and 2018 (Vision of Humanity, 2019). This shift has spurred numerous analyses of online behavior exhibited by lone actor terrorists. One such study, commissioned by the U.S. Department of Justice, identifies a common sequence of features associated with these actors' pathways, which include personal and political grievances, affinity with online sympathizers, identification of an enabler, broadcast of intent, and a triggering event (Hamm & Spaaij, 2015). A clear understanding of this process is crucial for detecting and preventing lone wolf attacks, particularly if the signs are discernible.

In this master's thesis, the focus is narrowed down to what could be considered to be a subset of lone wolf perpetrators, specifically school shooters. The last decade has witnessed a drastic rise in school shooting incidents, with the Violence Project's K-12 School Shooting Database (Riedman, 2023) reporting an alarming increase from 20 incidents in 2012, being right around the yearly average from $1970 - 2012$, to 304 shootings on school grounds in 2022 in the U.S. alone.

The emergence of the internet and social networking platforms has enabled the performers of these acts to engage with virtual communities of like-minded individuals, facilitating mutual radicalization and instruction on planning and executing attacks. Given the exponential growth of social media and the internet, it would be reasonable to assume that potential school shooters are among the increasingly large amount of users of these platforms. Possibly exhibiting early warning signs years in advance of an attack. Building upon this, it's important to note that potential threats may not always be explicitly spelled out in written posts. Instead, linguistic cues pointing towards an individual's propensity for violent behavior may be more subtle, embedded in their use of language.

With the number of school shootings steadily increasing, considerable effort has been invested in studying the psychological aspects of a school shooter. These studies mainly concern pedagogic approaches to prevent someone from ever becoming a school shooter. However, there are times when a person has already gone over the edge, and intervention is needed. Although over ten years old, previous studies revealed that every prominent school shooter between 2005 and 2010 had a presence on social media, with some leaving clues hinting to a potential future attack (Semenov et al., 2010). Studies on personality prediction and text classification have shown that a significant amount of information about a person can be learned from texts they have written (Park et al., 2014). With this in mind, it could be possible to leverage the information gained from analyzing texts to help in the fight against the increasingly large problem of school shootings. Traditionally this is done manually or through tips from students, and little work has been done to attempt to leverage machine learning to combat school shootings and lone wolf attacks (Neuman et al., 2020). As social media continue to grow, the data needed to be screened is increasing quickly, thus making automatic detection methods more relevant than ever before.

1.2 Goal and Research Questions

In light of the background and motivation outlined in the previous section, the following goal was formulated for this master's thesis:

Goal *Automatic detection of possible school shooters based on linguistic cues extracted from their written work.*

This thesis aims to use a quantitative method to make data-driven implementation choices to develop a solution to detect school shooters. Exploring multiple forms of features and machine learning techniques, the goal is to provide a solution for best encapsulating the context and linguistic cues deciding whether someone could be a potential school shooter. Three research questions to support this goal are described in detail below.

Research Question 1 *Which, if any, linguistic traits do school shooters have in common?*

This research question explores the traits and indicators connected to written posts by a school shooter compared to posts from non-shooters. The aim is to utilize the methods and finds from a preliminary study on lone wolf perpetrators, creating a base for comparison of the same techniques applied to posts by school shooters specifically.

Research Question 2 *How indicative are written records of an individual's potential to become a school shooter?*

This research question aims to investigate the feasibility of employing written records to assess an individual's propensity to commit a school shooting. By building upon the use of basic features, further incorporating word embeddings and state-of-the-art large language models, the goal is to ascertain the degree to which linguistic cues alone can predict the potential of an individual being a school shooter.

Research Question 3 *How suitable are machine learning methods for predicting the potential of someone performing a school shooting?*

The third and final research question invites an important dialogue concerning the ethical implications and practical applicability of employing machine learning models to identify potential school shooters. Given that the use of such models can directly affect humans, it is important to take into account the ethics tied to their application, in addition to assessing these models' scalability, reliability, and accuracy in real-world scenarios. Moreover, attention must be paid to the nature of the data used for training and prediction, addressing issues such as data availability, quality, and bias.

1.3 Research Method

To address the research questions and achieve the objectives of the thesis, multiple stages of research were undertaken. A literature review served as a vital precursor to the experimental work, ensuring the necessary knowledge about the prevailing strategies in the field. This process consisted of a structured literature review (Kofod-Petersen, 2018) complemented by a thorough exploration of references and citations from key publications using the snowballing approach (Wohlin, 2014), ensuring the discovery of the most pertinent works. Building upon the findings from this review, a quantitative approach was followed, taking what was considered to be promising solutions and experimenting with unique additions and adjustments.

The first step in this experimental approach was performed with Research Question 1 in mind. As both TF-IDF and LIWC features were found to yield promising results in previous studies, the same techniques were tested as part of the first experiment. To answer Research Question 2, the best solutions were determined by further experimentation, testing different combinations of features and algorithms, and ranking them based on the F_2-score as the primary metric. In addition, a facet of the qualitative research method has been applied, as our interpretation of the final model's performance on a smaller sample size is used to substantiate the answer to the question. The results of the experiments lay the foundation for answering Research Question 3, as well as the final discussion and conclusion in Chapter 7 and Chapter 8.

1.4 Contributions

- A labeled dataset containing 3028 unique texts written by 26 different school shooters from around the world.
- A voting classifier proficient at differentiating between texts written by school shooters and texts not written by school shooters.
- A discussion of the ethics and applicability of machine learning based approaches to screening of school shooters.

1.5 Outline

The remaining chapters of this thesis are organized as follows:

Chapter 2 presents the relevant background theory needed to understand the technologies used in this thesis and related work. The chapter details the methods, models, and metrics used in machine learning and natural language processing, in addition to a brief introduction to relevant social networking platforms.

Chapter 3 presents the findings of the conducted preliminary literature review, covering related work relevant to this thesis.

Chapter 4 will describe the data used, with examples and statistics for each dataset, in addition to a presentation of the preprocessing steps performed.

Chapter 5 presents the architecture of the thesis, from extracting features to training and testing the different types of machine learning models.

Chapter 6 describes the conducted experiments, including the experimental plan, setup, and results.

Chapter 7 evaluates the thesis research process and discusses the experimental results in light of the proposed research questions and goals. Furthermore, the discussion elaborates on the ethical considerations and usability of the resulting solution.

Chapter 8 provides a conclusion on the thesis and propositions for future work.

2. Background Theory

This chapter provides a comprehensive review of the relevant theory that underpins the research conducted in this thesis. First, the chapter begins with an introduction to the social media platforms associated with the datasets being used. It then delves into the key facets of Natural Language Processing (NLP), complemented by a section that explores Machine Learning (ML) models and techniques pertinent to classification and regression tasks on text-based inputs. Moreover, a presentation of the various evaluation metrics relevant to classification and regression follows. The chapter concludes with a display of the several technical tools that have been used in the performed research. Notably, a significant portion of this chapter was initially composed for a preliminary study and has been repurposed with necessary rewrites and adjustments where required.

2.1 Social Media Platforms

The widespread adoption of social media platforms has brought billions of people together, transforming how we connect as they enable instant communication and foster connections across geographical divides. At the same time, they've shaped our personal identities, influenced our behaviors, and opened new avenues for self-expression, learning, and social activism. These platforms have become a fundamental part of modern society and a rich user data and information source. This section will outline the platforms that hold the most relevance to the datasets used in the project.

2.1.1 Facebook

Facebook was launched in 2004 and has since become the most popular social media platform by users (Statista, 2023), with a reported number of daily active users slightly exceeding 2 billion in March 2023 (Meta, 2023). Each user creates their own profile, including information such as occupation, education, demographic information, hobbies, interests, and possibly a biography. On the main page, the user is met with a timeline of posts with the ability to react, comment, and share it either directly with their own list of friends or on their own user page. The posts can include a wide range of content, from plain or formatted text, images, videos, hashtags, and user mentions. In addition to interacting with friends through the feed, the social networking service lets the user create groups, events, and private messaging threads, further facilitating communication with both friends and strangers. As the platform has become increasingly popular over time, news outlets and organizations have taken to the platform, resulting in their published content populating each follower's feed as they "follow" them. Finally, the service employs targeted advertisement, utilizing massive amounts of collected data with the purpose of more efficient and personalized marketing.

2.1.2 Twitter

The microblogging platform Twitter joins the list of the largest social networks, with its reported monthly active users in January of 2023 hitting 556 million (Statista, 2023). Content-wise, the user is presented with a feed of posts, or "tweets", related to followed accounts, along with a thread dedicated to discussing or commenting on the post's content. Each post is strictly limited to 280 characters, which increased from 140 characters in 2017, and can include a variety of content such as plain text, media, hyperlinks, user mentions, and hashtags. The content on the platform can be viewed by anyone, even without creating an account. However, only registered users can post, comment, like, or reshare posts.

2.2 Natural Language Processing

Natural language processing is a field combining linguistics, computer science, and artificial intelligence, where the goal is to be able to process and interpret natural language. With an often ambiguous, unstructured, creative, and redundant nature, it differs from logical and structured languages such as those used in mathematics and programming. To be able to use the vast amounts of data that are collected every day, this ambiguous language needs to be translated into something that a computer can understand and utilize. This is where text processing, text representation, and feature selection come in. In this section, some of the most relevant techniques related to the processing of text within a document, and the representation of a collection of documents, often referred to as a corpus, will be described before elaborating on the feature selection task.

2.2.1 Fundamentals of Text Processing

Preprocessing of textual data is the act of structuring and sanitizing textual data prior to use in downstream tasks. Depending on the task at hand, the steps for preprocessing vary. Common preprocessing steps for most NLP tasks are:

- *Segmentation*: Extracting sentences from the document.
- *Tokenization*: Dividing the sentences into meaningful semantic units, often words or sub-words, called tokens.
- *Normalization*: The act of converting terms into a common canonical representation. In its simplest form, the process can include replacing non-alphabetical units such as numbers with the textual alternative, or transforming the casing to, for instance, lowercase.
- *Stemming*: Another form of text normalization involving the removal of word affixes, leaving only the word stem. This method needs careful application since certain words like "work", that function both as a noun and a verb, will be reduced to the same stem.
- *Lemmatization*: A more complex type of normalization, trying to group together the inflected forms of a word. Using the word's lemma or dictionary form, lemmatization enables each word, despite its inflection, to be analyzed as a single item. This is especially useful for cases where words have different stems in singular and plural

form, e.g., “foot” and “feet”, where stemming will take an unsatisfactory decision of not combining these into the same unit.

- *Stop word elimination*: Removal of frequently used words such as “a”, “for” and “the”. These words tend not to contribute to the overall semantics of the text and can therefore be removed. However, this should be performed with caution, as stop words are domain-specific, and some words often being listed as a stop word, such as “not”, can change the entire meaning of the sentence if removed.
- *Noise removal*: In certain domains, the use of elongated words, hyperlinks, user mentions, hashtags, misspellings, emojis, and other unconventional characters can be classified as noise. Because of the highly domain-specific nature of these instances, there might be some cases where removing these is seen as beneficial, but one should be aware of the possible loss of semantics.

2.2.2 Text Representations

One of the fundamental problems in text mining, information retrieval and natural language processing, is how to numerically represent the unstructured content in documents to make them mathematically computable. These representations can further be fed into machine learning algorithms as features. Some of the commonly used textual representations will be presented in this section.

TF-IDF

The Term Frequency-Inverse Document Frequency (Spärck Jones, 2004) is a numerical measure of a term's importance within a corpus. Consisting of two parts, the first component, Term Frequency (TF), measures the number of occurrences of a term within a single document. Multiple versions of the weight exist, but the most commonly used is either the count alone (raw frequency) or the log normalized version $1+log(\text{tf}_{d,t})$. The latter tends to be preferred because it makes them directly comparable to the IDF. Given the fact that an infrequent term is more distinguishing than frequent terms, it can be beneficial to highly rank the rarest terms, that is, those occurring in the least amount of documents. This can be achieved by taking the inverse of the Document Frequency, representing the number of documents in which a term occurs. This yields the second component of TF-IDF, being the Inverse Document Frequency (IDF). With the IDF commonly represented as $log(\frac{N}{\text{df}_t})$, the components can be combined into the equation below:

$$\text{TF-IDF} = (1 + log(\text{tf}_{d,t})) * log(\frac{N}{\text{df}_t})$$

Bag of Words

If the order or relationship between words is insignificant, the Bag of Words (Harris, 1954) representation can be a decent choice. The method keeps track of the words that appear in a document and expresses them as a vector with the same size as the vocabulary. Using either one-hot encoding or the term frequency, the Bag of Words (BOW) approach performs effectively in fields where simply the presence of words represents the content of a document.

N-Grams

N-grams are a way of representing text as an n-length sequence of words. By iterating word by word and combining n words together, the n-grams preserve the textual order to some degree for values higher than 1. Commonly applied representations are unigrams of one word, bigrams of two, or trigrams of three words.

Linguistic Inquiry and Word Count (LIWC)

The LIWC library, short for Linguistic Inquiry and Word Count, is a language analysis tool for analyzing text and counting the frequency of words that fall into pre-defined categories. These categories are based on the research and theory of Pennebaker et al. (2001), who identified words and language patterns indicative of various psychological and linguistic processes. The library includes a dictionary of words coded into various categories. For example, the category "affect" is based on the presence of words associated with positive or negative emotions, such as "happy" or "sad". Furthermore, the social processes category includes words related to social behavior, such as "friend" or "share". By feeding texts or documents into the LIWC software, the program calculates the percentage of words in each category before utilizing the distribution to derive semantic meaning. The output is a set of scores that indicate the relative prevalence of different categories of words in the text. This can further be used for classification and detection, as an attempt at personality modeling can be made based on the distribution of each category.

The licensed 2022 version of the software accommodates four dictionaries, encompassing the original 2001 version, along with subsequent updates released in 2007, 2015, and 2022. Each new version includes additional word categories that help to provide more nuanced insights into the content of the text. For example, the 2007 version extends the categories related to cognitive processes and social behavior with the scores for "insight", "friends", "social", and "humans". Other additions are categories covering the author's use of swear words, emoticons, and time-related verb forms. Furthermore, the 2015 version added new categories related to informational features, affective processes, and moral concerns. Some examples are the categories "clout", "authentic", "tone", "compare", "risk" and "big words". Finally, the 2022 version further expands the dictionary with categories related to health-related processes ("mental", "wellness", "fatigue", and "cognition"), motivation/drive ("need", "want", "allure", and "curiosity") and social/interpersonal behavior ("conflict", "moral", "politic", and "comm"). For explanations on the mentioned categories, refer to Appendix E. By incorporating these additional word categories, each new version of the software provides researchers with more powerful tools for understanding a text's emotional, cognitive, and linguistic content, allowing more fine-grained analyses of language use across different contexts and time periods.

Word Embeddings

Another technique for representing text is word embeddings, which aim to group together vocabulary words with semantic similarity. Moreover, embeddings address the problems associated with simpler methods' sparse vectors, inability to handle unidentified words, and lack of contextuality. Several implementations exist, such as Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), FastText (Bojanowski et al., 2016), and BERT (Devlin et al., 2019).

The first of these, Word2Vec, transforms words into a vector representation using either Continous Bag of Words (CBOW) or Skip-Grams. The CBOW method learns word embeddings by using surrounding words (context) to predict the current word as it slides a window across the sentences. Conversely, Skip-Gram learns to anticipate the context words given a specific word as input. In other words, these are opposite approaches to performing the same task. Global Vectors for Word Representation (GloVe), on the other hand, is an unsupervised approach that builds a co-occurrence term frequency matrix across the corpus to capture the global context. It then uses this matrix to calculate the similarity between words and represent them as vectors in a high-dimensional space. The resulting word vectors capture the semantic relationships between words based on their co-occurrence patterns in the text. The Fast Text Encoding using a pre-trained Character-level Model (FastText) implementation differs from the former by using subwords of length n or single characters ($n = 1$) as a base. As the subword might have been observed even though the full word was not present in the training data, the model might be able to predict rare or unseen words correctly. Finally, Bidirectional Encoder Representations from Transformers (BERT) uses a transformer-based architecture to generate embeddings considering a word's left and right context in a sentence. BERT is pretrained using a masked language modeling objective and a next sentence prediction objective on a large corpus of text, which allows it to capture a wide range of contextual information. The language model generates a fixed-length embedding for each token in a sentence, and its ability to capture contextual information and semantic relationships between words has made it one of the most potent pretrained language models available today.

Whether using general pretrained word embeddings or domain-specifically trained ones, their size is determined by the vector space's dimensions. Simply put, a larger vector space can hold more data. However, the performance is not necessarily proportionate to the size of the word embeddings, and a larger vector space does not always result in better performance. Hence, the challenge at hand should be considered when choosing the architecture, training method, and size of the word embeddings.

2.3 Machine Learning

Machine learning is a field within Artificial Intelligence (AI), which gives systems the ability to automatically learn and improve based on experience that has not been explicitly programmed. To train and adapt models for the generalization of data, which is the main essence of machine learning, one must first have a set of observations to explore potential underlying patterns. What is done next to find these patterns depends on the type of problem and data available, hence which subcategory of machine learning one is dealing with. It is common to divide machine learning into three categories: supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning aims to construct a model that predicts the correct label on unseen data. To achieve this, a function (model) is trained using prelabeled training instances, which is used to learn the relationship between the inputs (features) and the output (target label). This learned relationship enables the model to predict the labels of unseen data. Common examples of supervised learning are classification, where the model maps the input to a discrete set of predefined classes, and regression, where the input is mapped to continuous numerical values (Goodfellow et al., 2016, p. 100–101).

Unsupervised learning represents an exploratory approach to machine learning where the models work with unlabeled data instances. Without a target class, the objective shifts to discovering the underlying structures or associations in the data. Models in this category typically group data based on certain features, making it a frequent choice for cluster analysis with large volumes of data. Unsupervised learning can also be utilized in anomaly detection or to reduce data dimensionality through techniques like principal component analysis (PCA).

Reinforcement learning involves an agent that learns to perform the correct actions in a given *environment* or *action space* through a process of trial and error. The agent is typically rewarded for correct actions and penalized for incorrect ones. The goal of the agent is to learn a policy, a strategy that dictates the optimal action to take in each state in order to maximize its cumulative reward over time. This process gradually guides the agent toward its ultimate objective.

Given that the objective is clear — to identify a specific target class — and there's labeled data available to aid this process, this situation is an example of a supervised learning problem. The following discussion will cover the most prevalent techniques and methodologies within the domain of supervised machine learning.

2.3.1 Linear Regression

Linear regression is a supervised machine learning algorithm with the aim of fitting a line to a set of data points. Hence, a linear regression model is given by the equation of a straight line:

$$y = ax + b$$

Depending on the nature of the data, linear regression can be adapted to suit various types of problems, typically achieved by selecting an appropriate cost function. The most frequently used cost function, Mean Square Error (MSE), calculates the squared difference between the predicted value ($\hat{y}$) and the actual data point (y), essentially summing up these squared errors. By minimizing this function during training, the model can more closely align each data point to the optimal value. For datasets with noise or limited size, a too-complex linear regression model may overfit the data, leading to increased variance when the model is applied to new data. Extensions of linear regression, such as ridge regression and LASSO, address this issue. Ridge regression introduces a small amount of bias into the model by adding a penalty term to the cost function that shrinks the coefficients, thereby reducing variance. LASSO, on the other hand, not only shrinks coefficients but can also drive some of them to zero, effectively performing feature selection. These adaptations can lead to more generalized models compared to overfitted linear regression models.

2.3.2 Logistic Regression

Logistic regression establishes a relationship between input features and a binary outcome by fitting a logistic curve to the data, which makes it ideal for binary classification tasks. An example could be predicting whether a piece of writing is written by a native speaker or not only based on the presence of grammatical errors. Given a count of grammatical errors as input, the model then outputs a probability representing the likelihood of the text being written by a non-native speaker. This relationship would be learned through training on a labeled dataset, where each text sample is associated with its author being

a native or non-native speaker, enabling the model to understand patterns and make accurate predictions for new, unseen samples. If the x-axis in Figure 2.1 represents the number of grammatical errors in a text, the curve can be interpreted as the probability from $0-1$ on the y-axis of the text being written by a non-native speaker. The blue dots on the graph represent the data points used to fit the logistic regression curve.

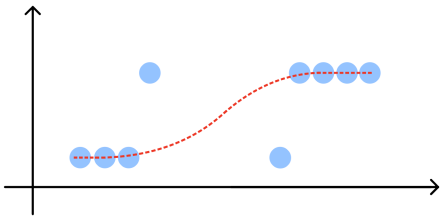


Figure 2.1: Logistic regression

2.3.3 Naïve Bayes

Naïve Bayes is a probabilistic machine learning classifier, acting as a simple implementation of Bayes' theorem to statistical data. The two most commonly used forms of Naïve Bayes are the Gaussian Naïve Bayes (GNB) and the Multinomial Naïve Bayes (MNB). The selection between these two depends on the nature of the data under study. For categorical data, the multinomial variant is the preferred choice, while Gaussian is more suitable for continuous data. In the field of Natural Language Processing, MNB is commonly employed to categorize terms or chunks of text utilizing a Bag of Words approach. This classifier determines the likelihood of the given data being part of each class defined in the classification task. However, in cases where the data is continuous, GNB ascertains the mean and variance for the samples within a given class. The likelihood of a specific feature value within a class is subsequently computed utilizing the Gaussian probability density function based on the previously determined mean and variance.

Regardless of which implementation you use, the probability of a feature A belonging to class B is given by Bayes' theorem:

$$P(B|A) = \frac{P(A|B) * P(B)}{P(A)},$$

where $P(B|A)$ represents the "posterior" probability, or the probability of class B given feature A. The numerator, $P(A|B) * P(B)$, is the product of the likelihood of observing feature A given class B and the "prior" probability of class B. Meanwhile, $P(A)$ is the total probability of feature A across all classes. This term serves as a normalization factor, ensuring that the sum of the probabilities of all classes given feature A is 1, which is important to make the calculated probabilities represent a valid probability distribution.

Finally, it's important to note that in probabilistic machine learning classifiers like Naïve Bayes, the "naïve" assumption of feature independence is made. This significantly simplifies the calculation of $P(A|B)$, as it becomes the product of individual feature probabilities given the class. This could be considered a limitation, as this independence assumption is often violated in real-world data, where features may be correlated or dependent. However, despite its assumption of independence, Naïve Bayes is still considered a decent classifier, even with strong dependencies between features (Zhang, 2004).

2.3.4 Support Vector Machine

The method of Support Vector Machine (Cortes & Vapnik, 1995; Hearst et al., 1998) builds on basic maximal margin classifiers. A maximal margin classifier is a classifier that tries to fit a hyperplane in such a way that the data points on either side and closest to the hyperplane have the largest possible distance to the hyperplane (Figure 2.2).

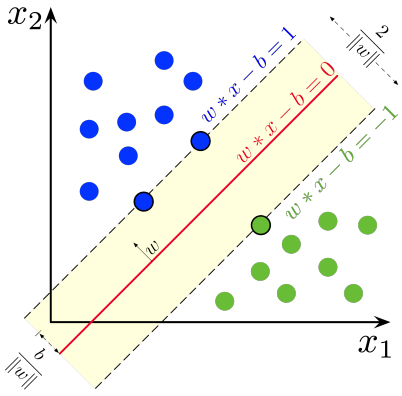


Figure 2.2: Support Vector Machine[1]

However, this classifier falls short when encountering outlier data. Imagine a case in one dimension where class 1 usually resides around the values $0-2$ and data points belonging to class 2 reside around the values $6-8$. At this point, a maximal margin classifier will function well since the classes are well separated. The problems start if we introduce data points belonging to class 1 that are much closer in value to the data points of class 2 than 1 (Figure 2.3). This outlier will force the maximal margin classifier into creating a separating threshold much closer to the values of class 2 than 1, leading to newly sampled points closer to class 2, possibly being classified as 1 instead!

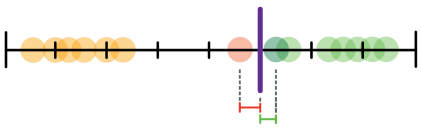


Figure 2.3: Maximal margin model weakness

The Support Vector Machine architecture tries to alleviate this problem by introducing two new approaches. These are support vectors and kernels. Support vectors allow for misclassification to benefit the larger amount of classifications (soft margin). Kernels transform data points to higher dimensions to allow for the classification of data not clearly split into two separate parts. Support vectors are vectors that run along the extreme data points closest to the maximal margin vector in parallel. The actual maximal margin vector is now seen as more of a soft margin since support vectors allow for some misclassifications to benefit the larger classification problem as a whole. The final vector is found by solving an optimization problem.

[1]Illustration from Wikimedia Commons, distributed under a CC BY-SA 4.0 license. Retrieved 20th May 2023, from: https://commons.wikimedia.org/w/index.php?curid=73710028

2.3.5 Decision Trees

Decision trees are a common type of machine learning algorithm. Some of their main advantages are the simplicity of calculation and the ease of interpretation, in addition to the applicability for both classification and regression problems. A decision tree consists of a set of nodes. Each node can be an end node, a branch node, a leaf, or a splitting node. At each splitting node, a comparison is done that splits the data further. The splitting variable or value is determined by what yields the highest information gain. This information gain is calculated by comparing the current node's *Gini impurity* to the potential total impurity of the nodes resulting from an eventual split.

The Gini impurity is given as:

$$Gini = \sum_{i=1}^{J} p(i) * (1 - p(i)),$$

where $p(i)$ is the likelihood of picking a data point from a class in the given node. Now, let U be the total information gain of a split, G be the Gini impurity score at a node, N be the total amount of data points, and n be a subset of the total data points. We can then express the information gain of a split as:

$$U = G_{\text{start}} - (G_{\text{left}} * (\frac{n_{\text{left}}}{N})) - (G_{\text{right}} * (\frac{n_{\text{right}}}{N}))$$

This splitting continues down the tree until a set stop condition is met. Common stop conditions are either when splitting no longer achieves satisfactory information gain or when a sufficient degree of certainty in classification has been reached in a given node.

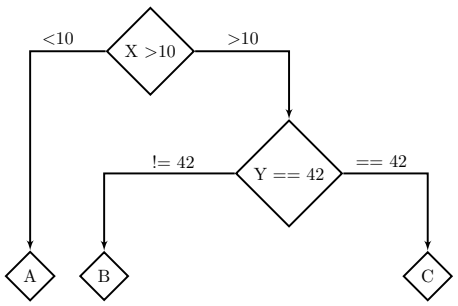


Figure 2.4: Decision tree

2.3.6 Ensemble Learning

Ensemble learning is a technique used in machine learning to improve the performance and accuracy of predictions by combining multiple individual models. This approach can be compared to making a new meaningful purchase, where usually multiple sources are consulted for opinions. The final decision can be made by considering the various evaluations, often through a majority vote. Ensemble learning is a versatile technique that can be used for various tasks, including classification, regression, and anomaly detection. Additionally, it can be combined with other machine learning techniques, such as deep learning and neural networks, to enhance its capabilities further. For example, a neural network can generate features before feeding these into an ensemble of classifiers to make more accurate predictions.

Two popular types of ensemble learning are boosting and bagging. The first of these, boosting, involves iteratively training a sequence of weak models to form a strong final model. Boosting algorithms assign more weight to misclassified examples in each iteration, emphasizing difficult cases. The final prediction is then made by combining the outputs of all the weak models. The XGBoost algorithm (Chen & Guestrin, 2016) is a popular example of a boosting algorithm. Another type of ensemble learning is bagging, which involves training multiple models on different subsets of the training data and then combining their outputs. This technique reduces the risk of overfitting and improves the overall accuracy and stability of the final model. Random Forests (Ho, 1995; Breiman, 2001) is a popular example of a bagging algorithm, being widely used in numerous real-world applications. Random Forests and XGBoost are examples of ensemble learning algorithms using decision trees as their base models.

2.3.7 Gaussian Processes

Gaussian processes (Rasmussen & Williams, 2005) is a kernel-based supervised machine learning algorithm. Adopting a probabilistic approach, it generates a probability distribution over output values by fitting a kernel function. Determining the similarity between input values or vectors in the input space, the idea is for the kernel function to output similar outputs on inputs close in space. Due to its ability to model complex nonlinear relationships in data and provide uncertainty estimates, the technique is a popular choice for a multitude of machine learning tasks. In addition, it offers a flexible approach to model selection, enabling easy integration of domain knowledge and prior beliefs.

One of the main advantages of Gaussian processes is its ability to perform Bayesian inference, which quantifies the uncertainty in predictions. This is particularly useful in applications where it is important to make decisions based on the reliability of the predictions. Additionally, Gaussian processes can handle missing data in a natural way, as the model can be trained on the available data and still make predictions on the missing data points.

However, the computational complexity of Gaussian processes increases with the number of observations, making them computationally expensive. Nonetheless, several techniques can reduce the computational cost, such as using sparse approximations or approximating the covariance function. Despite these challenges, Gaussian processes remain a powerful tool for modeling complex relationships in data with uncertainty estimates present.

2.3.8 Kernels

Support Vector Machines and Gaussian Processes are examples of linear machine learning algorithms that utilize a kernel function, also known as the *kernel trick*, to handle non-linear problems. A kernel function projects data points into a higher-dimensional space, allowing linear classifiers to separate non-linear data effectively.

Different kernel functions, such as the linear, sigmoid, dot product, white, and Radial Basis Function (RBF) kernel, can be applied based on the requirements of the task at hand. The linear kernel and the dot product kernel, which are quite similar, compute the dot product of two feature vectors. This process essentially measures the cosine of the angle between the vectors, capturing their directional similarity. The sigmoid kernel introduces a non-linearity that mirrors the activation function used in neural networks.

It applies the formula $tanh(\gamma x^T y + c_0)$, where γ is the slope, x^T and y are the vectors, and c_0 is a constant. This transformation can provide a useful link between linear models and more complex neural models. On the other hand, the RBF kernel operates somewhat differently, calculating an "inverse" distance between two vectors. This kernel function uses the formula $exp(-\gamma||x-y||^2)$, assigning higher similarity to pairs of points that are closer together. In this equation, γ is a configurable parameter, and $||x-y||$ represents the Euclidean distance between vectors x and y. Lastly, the white kernel plays a crucial role when the data incorporates noise. It adds a noise component to the diagonal of the kernel matrix, ensuring its invertibility.

2.3.9 Neural Networks

In recent years, many of the top-performing artificial intelligence systems have resulted from a technique called deep learning. Though the name *deep learning* is of newer existence, the name is, in fact, a new name for an approach called neural networks, which have been going in and out of fashion for almost 80 years. The first mathematical model of an artificial neuron, built upon the ideas of Alan Turing, was proposed by McCulloch and Pitts (1943). However, the *McCulloch-Pitts Neuron* has some limitations, as it can only represent non-weighted boolean functions with a hand-coded threshold. Overcoming these limitations, the first real implementation saw the light of day fifteen years later in the form of a machine by F. Rosenblatt, the "Mark I Perceptron". It could be argued that the whole area of deep learning and neural networks build on the suggested model by Rosenblatt (1958), called the *Perceptron model*.

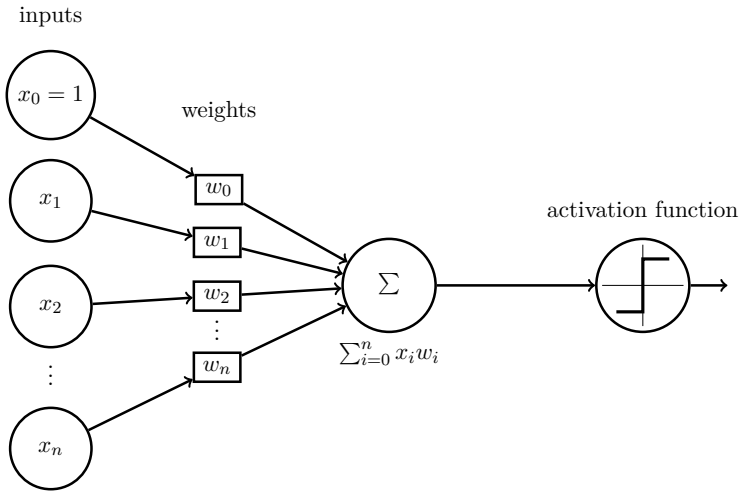


Figure 2.5: The perceptron model

Rosenblatt's perceptron model is illustrated in Figure 2.5. Provided with n input variables, here noted as x_1, x_2, and x_n, each entry is multiplied with a weight. This yields the main improvement, the introduction of a measure of importance, seeing as some of the inputs may weigh more than others on the decision of the desired outcome. The node then aggregates all the inputs and their belonging weights, taking the weighted sum before adding the bias. Instead of coding a threshold for the final step function, Rosenblatt adds a new weight with the same intention of shifting the activation function, but rather as an adjustable (and learnable) value.

The function in Figure 2.5 can be denoted as:

$$f(x) = \sum_{i=1}^{n} x_i w_i + b = \sum_{i=0}^{n} x_i w_i,$$

where the bias is included as an additional weight w_0, attached to a dummy input x_0 having assigned the value of 1. The output of this summation is then passed to a (Heaviside) step function to decide whether the neuron should fire.

In addition to improving the perceptron model with the addition of weights, a major achievement of Rosenblatt was to develop a fairly simple yet relatively efficient algorithm, enabling the perceptron to learn the correct synaptic weights from examples. This lays the foundation of Feature Learning, also known as Representation Learning, where the model can learn from a set of examples given as raw data.

Still, as pointed out by Minsky and Papert (1969), one limitation is the impossibility of computing XOR because it is not linearly separable. The paper argued that the proposed algorithm would not work as a model would need multiple layers, which the authors deemed too computationally expensive given the available hardware at the time. Although the potential is toned down, which is widely believed to have led the way to what is known as the first AI Winter, the later work on Multi-Layer Perceptrons, combined with the immense rise in computer power and available data, makes way for what is known as deep learning today.

Artificial Neural Network (ANN)

Artificial neural networks form the basis for many deep learning implementations. They can have vastly different structures depending on the problem they're designed to solve, but the simplest form is a feedforward neural network with layered nodes and weighted edges that allow for a unidirectional flow of information from input to output (Mitchell, 1997). A common distinction is made between two architectures based on the layers of nodes: networks with no hidden layers are known as Single-Layer Perceptrons, while those with multiple layers are referred to as Multi-Layer Perceptrons. Single-Layer Perceptrons are limited to linearly separable functions, whereas MLPs can handle high-dimensional data and overcome SLP limitations.

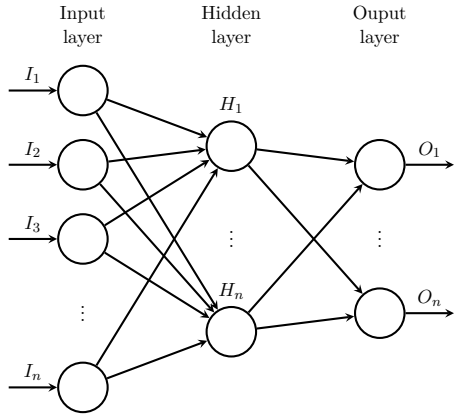


Figure 2.6: The simplest form of an Artificial Neural Network, the unidirectional implementation known as a Feedforward Neural Network

The neurons, also known as nodes, are in a MLP network layered into three or more layers, with a single input- and output layer and one or more hidden layers. Each layer passes its output, typically the product of the propagated information and the weight of the corresponding edge, as input to the next layer (O'Shea & Nash, 2015). Using an activation function, it could be determined whether to "activate" the node, thus controlling the propagation of information. The process of sending the data through the pipeline is called forward propagation or forward pass.

For learning to occur, that is, the adjustment of weights, a process called backward propagation or backward pass would need to be performed. This optimization technique compares the model's output to the correct output on labeled data, which yields an error measured as the distance between the correct and predicted value. Going backward, each weight is adjusted based on the derivative of the error with respect to the weight, also called the gradient. The process also goes by the name of gradient descent, trying to decrease the error or loss. The training, being one cycle of the forward and backward pass, continues on new data until the network converges.

According to the Universal Approximation Theorem, a neural network with just one hidden layer can theoretically approximate any function, provided the layer is large enough (Goodfellow et al., 2016). This means that, given enough nodes, it's always possible to find a neural network whose output, $g(x)$, satisfies $|g(x) - f(x)| < \epsilon$ for all inputs, x. In simpler terms, the approximation will be satisfactory for every possible input. However, although the theorem states that one layer is sufficient, in practice, using multiple layers is common to learn complex patterns from data effectively. Care must be taken, though, as adding more layers can lead to overfitting if not properly regularized. The optimal number of nodes and hidden layers is typically determined through experimentation and fine-tuning and is heavily dependent on the specific dataset and problem at hand.

Convolutional Neural Network (CNN)

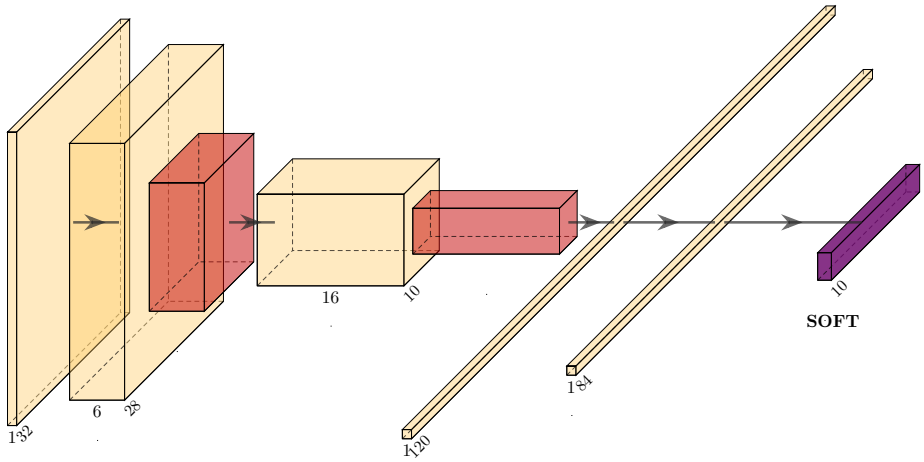


Figure 2.7: Architecture of a Convolutional Neural Network

Convolutional neural networks (LeCun et al., 1998) are a type of artificial neural network that have been particularly successful in solving computer vision problems such as image recognition and classification. They are a regularized version of the aforementioned MLP structure and are designed to exploit hierarchical patterns in data. As shown in Figure 2.7, being an illustration inspired by the architecture in the original paper by LeCun et al. (1998), CNNs are composed of an input layer, followed by a set of hidden layers resulting

in an output layer. Diverging from the MLP structure, the hidden layers in a CNN architecture incorporate convolutional and pooling layers before flattening the output into a vector and sending it through a series of fully connected layers. The convolutional layers act as filters on the input data, producing maps of activation that detect features such as edges, shapes, and textures. The pooling layers perform dimensionality reduction on the data by combining the outputs of neurons in a layer into a single neuron for the consecutive layer. In addition to dimensionality reduction, the pooling layer performs regularization to prevent overfitting and enable the layer to capture more local information.

The strength of CNNs is their ability to learn complex features from raw data hierarchically. By stacking multiple layers of convolutions and pooling, they can learn increasingly complex features, leading to better performance in tasks such as image classification and object detection. CNNs have also been successfully applied to Natural Language Processing (NLP) tasks, where single words can inherit important semantic meanings. CNNs are computationally intensive and require large amounts of data for training. However, the architecture has proven to be highly effective for many machine learning applications.

Recurrent Neural Network (RNN)

The recurrent neural network (Rumelhart et al., 1986) is a subcategory of artificial neural networks that can handle sequential data by allowing information to flow in cycles. Unlike feedforward neural networks, recurrent neural networks use each node's internal state to store information about previous calculations, allowing them to produce output based on past and current decisions. By assigning more importance to recent decisions, the network becomes more sensitive to immediate past events. This characteristic is advantageous when dealing with sequential data where recent events significantly influence current outcomes, such as the sentiment in the current sentence of a conversation being strongly influenced by the sentiment in the preceding sentence. The model architecture has been proven useful in many applications, such as speech recognition, language translation, and text analysis.

Long Short-Term Memory (LSTM) Network

The Long Short-Term Memory network is a type of recurrent neural network introduced in a paper by Hochreiter and Schmidhuber (1997). LSTM networks are designed to overcome the vanishing gradient problem, which occurs during backpropagation when the error gradients become increasingly small, causing information to be lost over time. In other words, they aim to overcome a significant limitation of RNNs — their inability to handle data with long-term dependencies, like long texts, due to limited memory capacity.

Long Short-Term Memory (LSTM) models utilize a memory cell to retain long-term data. The core structure of the LSTM cell includes three essential gates: the input gate, the output gate, and the forget gate. These components are visually represented in Figure 2.8, adapted from the original research paper (Hochreiter & Schmidhuber, 1997). The gates allow the network to selectively add or remove information from the memory cell based on the input data and the network's current state. The input gate determines what information from the input should be stored in the memory cell, while the forget gate controls what information shout be removed from memory. Finally, the output gate determines the information outputted to the next layer. By selectively storing and removing information, LSTMs can effectively process long data sequences while maintaining relevant information for future predictions.

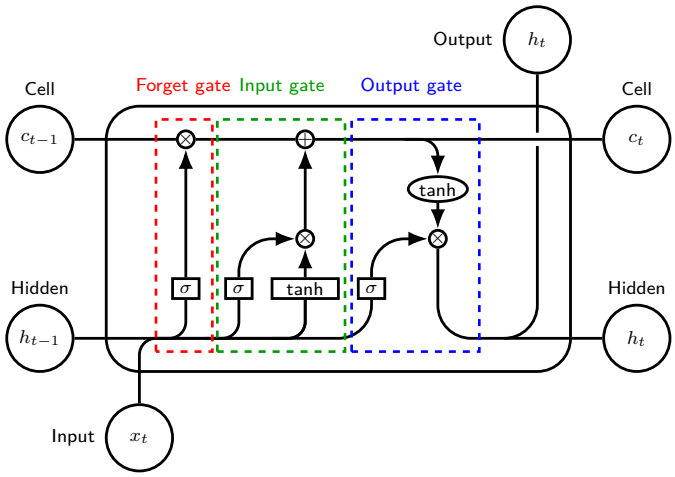


Figure 2.8: Visualization of an LSTM cell

LSTMs have been widely used in NLP tasks such as language translation and speech recognition, as well as in video analysis and image captioning. They have also been extended to include variations such as bidirectional LSTMs and stacked LSTMs, further improving their performance on various tasks.

The Encoder-Decoder Architecture and Attention Mechanism

The encoder-decoder architecture (Sutskever et al., 2014) is a prevalent concept in machine learning, particularly in the field of Natural Language Processing (NLP). This architecture is utilized to tackle problems where variable-length input sequences are transformed into variable-length outputs, such as machine translation, speech recognition, and time-series prediction.

The encoder-decoder architecture consists of two main components, as the name implies, the encoder and the decoder. The encoder processes the input data and converts it into a fixed-length vector representation, often called a "context vector". This vector holds the encoded information of the input, but given that the length of the vector is fixed, it is challenging to encapsulate long sequences without losing any information. The decoder uses this fixed-length vector to generate the variable-length output sequence. It aims to "decode" the information contained in the context vector, but it must interpret all of the input data from this single vector, regardless of the sequence length, which may lead to the loss of important details.

To help address these limitations, the attention mechanism was introduced (Bahdanau et al., 2015). This technique allows the model to focus on different parts of the input sequence when generating the output sequence rather than relying solely on a single context vector. In essence, it allows the model to "pay attention" to specific parts of the input that are more relevant at each step of the output generation. By doing so, it is capable of preserving a greater amount of the original information, enhancing its ability to manage sequences of varying lengths more effectively.

In the context of the encoder-decoder architecture, the attention mechanism computes a weighted sum of all input states, not just the final state, based on learned attention weights. These weights determine how much "attention" each input state should receive for each output step, helping to create a more comprehensive representation of the input.

The Transformer Architecture

Building on the concepts of the encoder-decoder architecture and the attention mechanism, the transformer model was introduced in a paper by Vaswani et al. (2017). This model revolutionized the field of Natural Language Processing (NLP) with its ability to handle long data sequences while maintaining computational efficiency.

The transformer model consists of an encoder and a decoder, but unlike the traditional encoder-decoder architecture, it relies solely on the attention mechanism (termed "self-attention" or "scaled dot-product attention") and entirely does away with sequence-aligned recurrent or convolutional layers. The transformer encoder reads the entire input sequence at once and generates a sequence of continuous representations for each word or token in the input. Each representation is a weighted sum of all input states, with weights determined by the attention mechanism. The attention mechanism allows each token in the input sequence to interact with every other token, enabling the model to capture long-distance dependencies between words. The transformer decoder also employs the attention mechanism but slightly differently to maintain the auto-regressive property (i.e., generating one word at a time). It has an additional "masked" self-attention layer that prevents positions from attending to subsequent positions, ensuring that the prediction for a particular position can depend only on known outputs at positions less than it.

A key feature of the transformer architecture is that it allows for parallelization during training, which leads to speed improvements compared to sequential models like RNNs or LSTMs. Furthermore, the transformer model includes several layers of these self-attention mechanisms, allowing it to learn more complex patterns and making it particularly effective for many NLP tasks.

BERT

Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based model introduced by Devlin et al. (2019). It revolutionized the field of natural language processing by demonstrating the power of unsupervised pretraining followed by task-specific fine-tuning. This approach led to state-of-the-art results across various NLP benchmarks, surpassing previous models. The key innovation of BERT lies in its bidirectional context representation. Unlike previous models that focused on either left-to-right or right-to-left context, BERT processes input text in both directions simultaneously, thereby capturing context more effectively. This bidirectional understanding allows BERT to understand the context of a word based on all of its surroundings (left and right of the word), making it highly effective for a range of NLP tasks.

BERT is pretrained on two unsupervised tasks, namely Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). The MLM task randomly masks a percentage of the input tokens from the input sequence to predict, encouraging the understanding of the sentence context. NSP, on the other hand, trains the model to understand relationships between sentences, which is particularly useful for tasks like question answering and summarization. Once pretrained, BERT can be fine-tuned for various tasks such as text classification, named entity recognition, and question-answering by adding a task-specific head on top of the pretrained model. This allows the model to transfer the general language understanding learned from pretraining to specific tasks, often leading to significant performance improvements with relatively little task-specific training data.

RoBERTa

Robustly Optimized BERT Pretraining Approach (RoBERTa) is an optimized version of BERT introduced by Liu et al. (2019). RoBERTa builds upon the success of BERT by fine-tuning the pretraining process, including changes in hyperparameters, model architecture, and the training data.

The main differences between the optimized version over the original pretrained transformer architecture of BERT are as follows. First, RoBERTa removes the Next Sentence Prediction (NSP) task from pretraining, focusing solely on Masked Language Modeling (MLM). Second, it also extends the pretraining with the use of more training data, larger batch sizes, and more iterations. Finally, the model uses dynamic masking patterns for the MLM task instead of static masking found in BERT. These optimizations improve performance across various NLP tasks, surpassing BERT's already impressive results on certain applications.

DistilBERT

DistilBERT (Sanh et al., 2020), or the Distilled Version of BERT, is a condensed yet highly efficient version of BERT, designed to maintain most of BERT's performance while minimizing model size and computational demands. The main idea is to use what is known as knowledge distillation, a technique that involves training a smaller model, the "student", to mimic the behavior of a larger, pretrained model, namely the "teacher". In the case of DistilBERT, as the name implies, BERT serves as the teacher model. The key differences between the two models lie primarily in their structure and performance. DistilBERT typically has half the number of layers compared to its teacher, BERT. In addition, it does not include token-type embeddings or the pooler layer, which in BERT is used to summarize the context of the input sequence. These modifications make DistilBERT approximately 40% smaller and 60% faster than BERT. Despite these reductions, DistilBERT remarkably retains around 97% of the performance across a variety of NLP tasks, according to the paper by Sanh et al. (2020).

ALBERT

A Lite BERT (ALBERT) is another optimized version of BERT, introduced by Lan et al. (2020). Similar to the Distilled Version of BERT, this variation focuses on reducing the model size while maintaining similar performance levels.

Two main innovations are being introduced. The first is *factorized embedding parameterization*, where the model separates the size of the hidden layers from the size of the input embeddings, allowing for a significant reduction in the number of parameters without sacrificing performance. Secondly, ALBERT shares parameters across layers in the transformer architecture, reducing the overall number of parameters and the memory footprint. This is called *cross-layer parameter sharing*. These modifications result in a much smaller and more memory-efficient model than BERT, without significantly losing performance. ALBERT has also been pretrained on larger-scale tasks and can be fine-tuned for various natural language processing tasks similar to the rest of the aforementioned pretrained transformer architectures.

2.4 Evaluation Metrics

To be able to determine whether machine learning models reach their goal of either regression or classification, one needs to have a measure of performance. This section describes some of the evaluation metrics pertinent to these tasks. For classification problems, this includes the metrics of accuracy, precision, recall, and F-Score, while some forms of error are covered because of the use in regression tasks.

2.4.1 Accuracy

The first performance-related metric, accuracy, is used in a wide variety of machine learning tasks. By quantifying the number of correctly classified instances, divided by the total number of instances, a sense of how many correct classifications the model makes could be gained, thus also knowing the number of misclassifications. The metric is denoted by the following equation:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

Upon examining the equation, it becomes apparent that it comprises four separate values, namely TP, TN, FP, and FN. True Positive (TP) is the number of correctly classified positive instances, while True Negative (TN) represents the instances that are correctly predicted as negative. In contrast, False Positive (FP) refers to the negative instances incorrectly classified as positive, and False Negative (FN) represents the positive instances that are misclassified as negative. These four values can be better visualized in a tabular form, as seen in Table 2.1.

		Ground Truth	
		Positive	Negative
Prediction	Positive	TP	FP
	Negative	FN	TN

Table 2.1: Matrix illustrating the four outcomes in binary classification

2.4.2 Precision and Recall

Another performance metric commonly used in classification problems in both information retrieval and machine learning is precision and recall. Precision, or the positive predictive value, is the portion of relevant instances among the retrieved instances. To elaborate, the measure tells us *how many of the returned items are relevant.* The equation for precision can therefore be presented as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$

where the elements True Positive (TP) and False Positive (FP) are the same as presented in the former subsection. Recall, alternatively termed sensitivity, quantifies the fraction of relevant instances successfully retrieved. Contrary to precision, which primarily concerns itself with the correctness of the instances retrieved, recall lays emphasis on the volume of

relevant instances recovered from the total set of elements or documents. In other words, it assesses the ability to *retrieve as many relevant instances as possible*, disregarding the correctly classified instances within the retrieved subset. The mathematical representation of recall is expressed as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

2.4.3 F-score

The F-score attempts to be a measure of accuracy, including both precision and recall. In its simplest form, the F_1-score is the harmonic mean of the two given metrics. Furthermore, the more generic F_β-score implements an additional weight, β, valuing either precision or recall more than the other. The output is a score on a scale between 0 and 1, where 1 indicates the ideal precision and recall. The equation for the more general F_β-score can be written as:

$$\text{F}_\beta = (1 + \beta^2) \times \frac{\text{Precision} \times \text{Recall}}{(\beta^2 \times \text{Precision}) + \text{Recall}}$$

Setting β to 1 yields the harmonic mean between recall and precision. This is the standard F_1-score.

$$\text{F}_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Similarly, if β is set to 2, we get the equation for the F_2-score, weighing recall heavier than precision:

$$\text{F}_2 = 5 \times \frac{\text{Precision} \times \text{Recall}}{(4 \times \text{Precision}) + \text{Recall}},$$

2.4.4 Binary Cross-Entropy Loss

Binary Cross-Entropy Loss, also known as log loss, is a loss function commonly used in binary classification tasks with models that output probabilities. Given a model that predicts probabilities, the binary cross-entropy loss measures the dissimilarity between the true label (which can be either 0 or 1) and the predicted probability of the instance belonging to the positive class (usually denoted as 1). The formula for binary cross-entropy loss for a single instance is:

$$L = -[y * log(p) + (1 - y) * log(1 - p)],$$

where y is the true label (0 or 1), and p is the predicted probability of the instance being in class 1. A key property of this loss function is penalizing confident and wrong predictions more heavily than confident, right, or unconfident (i.e., probabilities close to 0.5) predictions. This property makes it suitable for training models on imbalanced datasets, where the penalty for misclassifying a minority class instance needs to be higher.

2.4.5 Error

As mentioned in the introduction, the former metrics are used in classification tasks while not being an applicable measure for regression problems. Instead, different forms of measuring error can be used to evaluate a regression model's performance. The three most commonly used measures are the Mean Absolute Error (MAE), Mean Square Error (MSE), and Root Mean Square Error (RMSE). The first of these, Mean Absolute Error, is a measure of the absolute average distance between the expected value and the value given by the model. Subsequently, Mean Square Error takes a similar route but measures the average square distance between the predicted and expected values instead. Finally, the Root Mean Square Error builds upon MSE by using its square root. This has the effect of penalizing larger errors more heavily than smaller ones, which is a useful feature.

2.5 Technical Tools

The section briefly describes the main technical tools employed in this thesis. Python, a high-level and versatile programming language known for its clear syntax and wide-ranging applications in data analysis and machine learning, is at the heart of the technological infrastructure. The tools selected for this study, including Pandas, NLTK, PyTorch, Scikit-learn, Ray Tune, and Hugging Face Transformers, among others, all interface smoothly with Python, contributing to a cohesive analytical framework.

Pandas (McKinney, 2010) is a Python data manipulation and analysis library. Its primary data structures, dataframes, and series are extremely flexible and efficient, making Pandas an essential tool for managing and processing large and complex datasets.

Natural Language Toolkit (NLTK) (Loper & Bird, 2002) is a Python library for working with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources, allowing for the preprocessing and analysis of textual data.

Scikit-learn (Pedregosa et al., 2018) is a Python module for machine learning built on top of SciPy. It provides a selection of efficient tools for machine learning and statistical modeling, including classification, regression, and clustering via a consistent interface. The inherent simplicity of this tool allows researchers to manage model training, validation, testing, and evaluation with minimal coding effort.

Ray Tune (Liaw et al., 2018), a component of UC Berkeley's Ray Project, is an open-source library designed for hyperparameter tuning. The library uses a universal API to define search spaces and leverages cutting-edge optimization techniques, supporting seamless integration with key machine learning frameworks.

PyTorch (Paszke et al., 2019) is a Python library for deep learning. The library offers a flexible and intuitive interface for constructing and training various machine learning models. Additionally, its TorchText module furnishes a robust suite of tools for processing, loading, and handling textual data, easing the pipeline creation for NLP tasks.

Hugging Face Transformers (Wolf et al., 2020) is a state-of-the-art library for Natural Language Processing (NLP) in Python. It provides thousands of pretrained models to perform text-based tasks, such as text classification, information extraction, and text generation. Transformer models have revolutionized the field of NLP, and the Hugging Face framework provides a flexible, efficient, and scalable way of working with them.

3. Related Work

In preparation for this thesis, an initial specialization project was carried out to gain a better understanding of the research areas of school shooter detection and personality modeling. This endeavor included a comprehensive literature review, the specifics of which will be detailed in the opening section of this chapter. Additionally, insights gleaned from this review, including pertinent datasets, models, and steps for preprocessing and feature extraction, will be repurposed and elaborated on within this chapter.

3.1 Literature Review

In the preliminary specialization project, two literature review methods were utilized, more specifically, a structured literature review and the snowballing approach. In this section, the use of both methods will be presented.

3.1.1 Structured Literature Review

The structured literature review was conducted based on a suggested outline by Kofod-Petersen (2018). This subsection will present the process, detailing the five steps implemented across the three phases of the recommended structure. The first phase, planning, involves specifying the research questions and developing a review protocol. The second phase encompasses the actual execution of the review process. This entails the identification of research, selection of primary studies, evaluation of study quality, extraction of data, progress monitoring, and synthesis of data. In the third and final phase, the objective is to communicate the newly acquired knowledge and should therefore involve a specification of a dissemination strategy, a formulation of a report, and an evaluation of the completed report. A detailed description of each step, along with the outcomes of the execution phase, are provided in the forthcoming subsections.

Step 1: Identification of Research

The process of collecting and exploring the literature was facilitated through Google Scholar[1], a widely recognized and extensively used academic search engine. This platform was chosen due to its comprehensive coverage of scholarly literature across various disciplines and formats. In an effort to identify literature specifically pertinent to the project's objective, two carefully designed search queries were crafted.

[1] https://scholar.google.com

Q1: `(school shooting OR gun violence OR shooting OR violent act) AND (machine learning OR computational OR ai OR prediction OR classification) AND (social media OR 4chan OR twitter OR reddit OR facebook OR gab OR weibo)`

The first query focuses on literature concerning the automatic prediction of individuals predisposed to violence who could potentially execute a school shooting.

Q2: `(machine learning OR computational OR ai OR prediction OR classification) AND (social media OR 4chan OR twitter OR reddit OR facebook OR gab OR weibo) AND (personality OR personality traits OR personality profiling)`

The second query tries to extract literature concerned with automatic personality profiling and prediction using data mined from social media. Both queries were constructed based on a set of key terms deemed highly relevant to the problem domain. The key terms are presented in Table 3.1.

Group 1: Violence	**Group 2: Machine Learning**	**Group 3: Social media forums**	**Group 4: Personality prediction**
School shooting	Machine learning	Social media	Personality
Gun violence	Computation	4chan	Personality traits
Shooting	AI	Twitter	Profiling
Violent act	Prediction	Reddit	
	Classification	Facebook	
		Gab	
		Weibo	

Table 3.1: Search terms for the conducting phase described in Kofod-Petersen (2018)

Step 2 + 3: Selection of Primary Studies

To keep the scope of the literature review manageable, the articles retrieved from the queries were limited to the first 50. However, before these were included, a set of predetermined exclusion criteria was applied. The criteria for removal included duplicate entries and studies published more than 10 years prior to our research period, in this case, before 2013. As a final step, the studies retrieved from the initial search were subject to a study quality assessment to ensure only relevant papers were included. The inclusion criteria and quality assessment criteria used can be found in Appendix A and Appendix B. Following this examination, a total of 12 papers were deemed suitable for inclusion. The studies retrieved from the structured literature review are presented in Appendix C.

Step 4 + 5: Data Extraction, Monitoring, and Synthesis

Finally, the data retrieved from the corpus resulting from the Structured Literature Review (SLR) is summarized in Table 3.2.

Table 3.2: Data synthesized from SLR

Id	Title	Author(s)	Year	Algorithm	Features	Dataset(s)	Summary
1	Identifying Warning Behaviors of Violent Lone Offenders in Written Communication	L. Kaati, A. Shrestha & T. Sardella	2016	Adaboost with classification trees	LIWC	Texts from schoolshooters.info, publicly available written comm. from mass shooters, posts from the white supremacist forum Stormfront, and other forums	Found differences in psychological features retrieved from LIWC for non-violent offenders vs. violent offenders. The differences in the categories "anger" and "psychological process" were important in separating offenders and non-offenders.
2	Profiling School Shooters: Automatic Text-Based Analysis	Y. Neuman, D. Assaf, Y. Cohen & J. L. Knoll	2015	KNN, Tree classification (CHAID), and binary logistic regression	Vectorial semantics approach used as features	Blogs Authorship Corpus and texts written by school shooters gathered from schoolshooters.info	Using a more statistics-based approach, the authors constructed a ranking of texts most likely to have been written by a school shooter. Using their method, the school shooters' texts were all contained in the first 210 texts of the ranking = 3% of their total corpus.
3	A Multi-Label, Semi-Supervised Classification Approach Applied to Personality Prediction in Social Media	A. C. E. S. Lima & L. N. Castro	2014	Naïve Bayes, SVM, and MLP	LIWC and MRC Psycholinguistic Database	Obama-McCain Debate, Sanders and SemEval2013	Accomplished an accuracy of around 83% on Big 5 personality traits.

Continued on next page

Table 3.2 – continued from previous page

Id	Title	Author(s)	Year	Algorithm	Features	Dataset(s)	Summary
4	Personality Predictions Based on User Behavior on the Facebook Social Media Platform	M. M. Tadesse, H. Lin, B. Xu & L. Yang	2018	XGBoost, SVM, logistic regression, and gradient boosting	LIWC, SPLICE, and Social Network Analysis (SNA)	myPersonality	Found that extroverted users tend to use fewer, but more positive words. XGBoost outperformed other models, most notably on the extroversion feature (78.6%). The SNA feature set outperformed traditional linguistic feature sets such as LIWC.
5	Personality Classification Based on Twitter Text Using Naive Bayes, KNN and SVM	B. Y. Pratama & R. Sarno	2015	MNB, KNN, and SVM	Vector space model	User data and posts retrieved from Twitter, myPersonality (translated to Indonesian)	Used a binary classifier for each class in the Big 5 Model. After tokenization, stemming, filtering of stop-words, and weighting, they achieved a max accuracy of 60% with MNB, ultimately failing to improve on previous work.
6	Recognising Personality Traits Using Facebook Status Updates	G. Farnadi, S. Zoghbi, M.-F. Moens & M. De Cock	2013	KNN, SVM, and NB	LIWC, Social Network Analysis (SNA) features (network characteristics and temporal features), and other content metrics	myPersonality and Facebook status updates	Used binary classifiers for multi-class classification of Big 5 Model. Found that there is no single kind of feature that give the best results for all Big 5 personality traits. They argue ML approaches to personality recognition are generalizable across domains.

Continued on next page

Table 3.2 – continued from previous page

Id	Title	Author(s)	Year	Algorithm	Features	Dataset(s)	Summary
7	Predicting Personality Traits of Chinese Users Based on Facebook Wall Posts	K.-H. Peng, L.-H. Liou, C.-S. Chang & D.-S. Lee	2018	SVM, NB, and logistic regression	Used BOW approach for features with TF and TF-IDF schemes	Facebook user data and posts, in addition to a short online survey created by the authors	Used Big 5 with SMOTE to overcome class imbalances. Achieves surprisingly high accuracies, with around 95%+ being common for all algorithms used.
8	Personality Prediction System from Facebook Users	T. Tandera, H. Derwin, S. Rini, W. Yen & L. Prasetio	2017	NB, SVM, logistic regression, gradient boosting, LDA, MLP, LSTM, GRU, CNN, and LSTM + CNN	LIWC2015, SPLICE, and SNA features for traditional methods, and deep learning with open vocab. (word embeddings - GloVe)	myPersonality + 150 manually sampled Facebook users	The authors compared a deep learning methodology with an open vocabulary approach using GloVe against traditional machine learning algorithms. Their findings reveal that, given sufficiently large datasets, deep learning strategies have the potential to surpass the performance of conventional machine learning techniques.
9	Mining Facebook Data for Predictive Personality Modeling	D. Markovikj, S. Gievska, M. Kosinski & D. Stillwell	2013	SVM, Simple Minimal Optimization (SMO), MultiBoostAB, and AdaBoostM1	Social Network features (e.g., friends or likes), demographic (e.g., age or gender), LIWC, Part-of-Speech (POS) tag, AFINN and H4Lvd from General Inquirer	myPersonality	The study confirmed prior results that there is no universally optimal set of features for all classes. However, the selection of more discriminative features was found to enhance accuracy.

Continued on next page

Table 3.2 – continued from previous page

Id	Title	Author(s)	Year	Algorithm	Features	Dataset(s)	Summary
10	Automatic Personality Assessment Through Social Media Language	G. Park, H. A. Schwartz, J. C. Eichstaedt, M. L. Kern, M. Kosinski, D. J. Stillwell, L. H. Ungar & M. E. P. Seligman	2014	Ridge regression	Open vocabulary features (words, phrases, and topics). Dimensionality reduction: Univariate feature selection and random PCA	77,556 users of myPersonality	Used the NEO-PI-R five factor model. The study applied ridge regression with the mentioned features to attain state-of-the-art performance. Concludes that open vocabulary captures more semantic meaning.
11	Personality Traits on Twitter or How to Get 1,500 Personality Tests in a Week	B. Plank & D. Hovy	2015	logistic regression classifier	Binary word n-grams and metadata (e.g., followers, number of tweets, statuses)	Tweets from Twitter users combined with self-assessed Myers-Briggs Type Indicators	Study found that linguistic cues are the strongest indicators of personality. Meta-features can help add to accuracy.
12	25 Tweets to Know You: A New Model to Predict Personality with Social Media	P.-H. Arnoux, A. Xu, N. Boyette, J. Mahmud, R. Akkiraju & V. Sinha	2017	Gaussian processes	Word embeddings (GloVe)	Tweets from 1.3k Twitter users surveyed via web-app, IPIP format	Word embeddings and Gaussian processes approach outperforms previous state-of-the-art (LIWC + ridge regression). Dense word embeddings improve performance on short texts.

3.1.2 Applying the Snowballing Technique

As described in Section 1.3, the snowballing approach was used to find additional relevant literature. Following the procedure described in Wohlin (2014), an initial start set of three documents was made using Google Scholar. The first among these was the study by Kaati et al. (2016), which was identified during the Structured Literature Review (SLR) and hence, is included in Table 3.2. Furthermore, the work of Neuman et al. (2020) was found in an author search from one of the initial literature review findings (Neuman et al., 2015). Finally, the paper by Simons and Meloy (2017) was included for trying to supplement with documents regarding threat assessment, adding a level of diversity as mentioned by Wohlin (2014). This particular paper was selected due to being a reference in Neuman et al. (2020), in addition to being a seemingly popular article and part of a highly rated book associated with the field. Using this relatively small start set, justified by the fact that the snowballing approach was intended to be supplementary to the SLR, both the backward and forward snowballing approaches were followed for each of the articles. The resulting papers, presented in Appendix D, can serve as additional literature to the work already presented.

In the following sections, the findings derived from both the conducted Structured Literature Review and the application of the snowballing method will be presented.

3.2 Datasets

Online platforms and social media provide vast amounts of data that can be leveraged for various Natural Language Processing (NLP) tasks. This has resulted in a substantial amount of work dedicated to gathering datasets from social media for personality profiling purposes. However, there appears to be a scarcity of datasets directly linked to profiling school shooters. This section will introduce the datasets considered to be relevant for further work in this area.

3.2.1 Personality Profiling Datasets

The large amount of data gathered from social media allows for an insight into people's lives and habits. Several datasets pertaining to personality profiling have been constructed based on social media activity. The use and creation of these datasets will be covered in the following subsections.

Facebook Datasets

Facebook is a platform that enables users to join groups, engage in chats, and post messages on various forums. One of its former applications, *myPersonality*, offered users a brief questionnaire that would generate a Big 5 personality score upon completion. The collected dataset incorporated user posts, their respective personality scores, and user-specific data (friends, groups, likes, etc.). According to the dataset's authors, the application had at least 6 million users complete the questionnaire, with about 40% of them agreeing to their data being used for research (Stillwell & Kosinski, 2015). The dataset was discontinued in 2012, and scholarly access to the full dataset was removed in April of 2018. However, the dataset is still perhaps the most used dataset linked to Facebook.

Tadesse et al. (2018) used a subset of 250 users, including 9917 status updates of the myPersonality dataset. Their final dataset contained user information, status updates, Big 5 personality labels, and Social Network Analysis (SNA) features. They attempted to improve on previous results by feeding SNA data into an XGBoost architecture. Pratama and Sarno (2015) retrieves a subset of 250 myPersonality users accompanied by 10000 user posts in total. These posts were then translated into Indonesian and appended with Twitter posts from the corresponding person. The authors assume that not too much meaning is lost in translation. The study done by Farnadi et al. (2013) opts to use the same amount of users as both Pratama and Sarno (2015) and Tadesse et al. (2018). A total of 9917 posts are used in combination with a corpus of essays annotated with personality traits collected by Mairesse et al. (2007). Their approach attempts to incorporate spatial data as well as user metadata to predict users' Big 5 personality scores. Tandera et al. (2017) compares the efficacy of traditional machine learning algorithms such as SVMs and modern deep learning architectures. Similar to others, they retrieved 250 users with 10000 status updates for their dataset. However, they further expanded the dataset by manually collecting data from an additional 150 users through the Facebook API, all of which were annotated with Big 5 personality scores. The largest study using myPersonality found in the literature review was by Park et al. (2014), using 71556 users from the dataset. In this study, ridge regression was applied with external personality tests used as validation.

Twitter Datasets

Twitter is another social media giant with millions of users active every day. As mentioned in Section 2.1, the social media platform is a microblogging site allowing users to write posts with at most 280 characters. The platform allows users to write what they want with little filter, making it easy to express their true feelings, whatever they may be. Several datasets have been created to take advantage of this quality of Twitter for personality prediction. Plank and Hovy (2015) uses a novel dataset of over 1.2 million tweets annotated with the Myers-Briggs Type Indicator (MBTI) personality types and gender to attempt to predict personality dimensions. The dataset was constructed by the authors and made available on a public code repository for others to use. It is worth noting that the MBTI types annotated are taken from the users' own Twitter account description. Arnoux et al. (2017) constructs a novel corpus from a self-made application. This dataset contains tweets from 1323 unique users annotated with their Big 5 personality traits. Lima and de Castro (2014) proposes a multi-label classifier approach using the Big 5 Model as class labels. For performance evaluation, they utilize different sentiment analysis datasets. The Obama-McCain Debate (OMD) dataset contains 3238 tweets from the 2008 presidential campaign annotated with sentiment scores. The tweets are accompanied by date and user identification. The SemEval2013 and Sanders Analytics datasets were also used in their study. The latter seems to be discontinued but has been backed up on GitHub (Sanders Analytics, 2013). Finally, an effort to combine Twitter data with user profiles found on Facebook has been done, as mentioned in the former section.

3.2.2 Datasets on School Shooters and Lone Wolf Perpetrators

With such a large amount of users active on social media every day, it is only natural to assume that also potential lone wolves or school shooters utilize them. As of the time of writing, there are no publicly available annotated datasets directly linking social media posts and accounts of acknowledged lone wolf perpetrators known to the authors.

Work done in this area has relied on the construction of novel datasets, such as the work of Ekwunife (2022). Ekwunife constructed a dataset consisting of 500 tweets related to five different mass shooting incidents in the U.S. Neuman et al. (2020) and Kaati et al. (2016) attempt to overcome this scarcity of social media data by using established databases of known school shooters. The database used was *schoolshooters.info* (Langman, 2022). This database contains an overview of registered school shooters in the U.S., along with documents relevant to each case. The documents used in both studies were written material produced by the school shooters prior to an incident.

3.3 Preprocessing and Feature Extraction

This section is set to describe the methodologies identified for preprocessing and feature extraction, specifically used in studies related to personality prediction and the identification of lone wolf perpetrators.

3.3.1 Preprocessing

Preprocessing is a crucial step of any NLP task; however, the amount and extent of the preprocessing vary depending on the task at hand. Due to the inherently noisy nature of social media posts, the preprocessing step usually has to weigh the amount of processing to be done up against the potential loss of semantic meaning. Park et al. (2014) choose to perform a minimal amount of preprocessing for this reason. They utilize the emoticon-aware tokenizer made by Potts (2011), which preserves punctuation. Palomino and Aider (2022) argue that the removal of punctuation could hurt sentiment analysis due to special characters or emoticons potentially conveying sentiment. The same paper stresses the importance of preprocessing and found that a combination of common preprocessing steps indeed does increase the accuracy of sentiment analysis on social media texts. Due to the concerns of over-processing listed above, the majority of papers discovered during the literature review adhere to standard practices, including lowercasing, hyperlink elimination, special character removal, and punctuation removal. On the other hand, Plank and Hovy (2015) found that the removal of stop words harmed performance when trying to model Big 5 personality traits. This emphasizes the importance of tailoring the preprocessing based on the specific dataset in use and the task at hand.

3.3.2 Feature Extraction

Features utilized for automatic detection and personality prediction typically fall into two primary categories: linguistic features and user-specific features. This section provides a brief overview of these features as leveraged in previous related studies.

Linguistic Features

Linguistic features are features that can be extracted from written text. Plank and Hovy (2015) found that in their study, the strongest indicator of personality was linguistic cues. Therefore, sufficient work should be done to extract the potential cues to ease the process of personality prediction. The following subsection presents the linguistic features employed in the studies identified through the literature review.

Linguistic Inquiry and Word Count (LIWC) have been widely used for linguistic cue extraction. LIWC is a large framework with fine-grained features that can contribute to personality prediction. However, not all features are discriminative enough, leading to the need for feature selection. Kaati et al. (2016) applied LIWC features to screen for lone offenders in written text. They found that using the Mahalanobis distance to rank each LIWC feature and selecting the 11 highest-ranking features resulted in an increase in accuracy. Important features were articles, prepositions, and negative emotions. From the work of Tadesse et al. (2018), the correlation between LIWC features and personality is further analyzed using the Pearson correlation coefficient. An interesting discovery is that features from the LIWC dictionary appear to be more discriminative or exhibit a stronger correlation with personality compared to the newer SPLICE dictionary.

Although older, the traditional statistical approach of Bag of Words (BOW) and TF-IDF weighting is still used as an important tool in personality prediction, often combined with Naïve Bayes classifiers. In the paper titled "Predicting Personality Traits of Chinese Users Based on Facebook Wall Posts", Peng et al. (2015) apply a simple TF-IDF weighting process to represent each user's posts as a vector. It is worth noting that the words given to the BOW model were tokens constructed by the *Jieba Chinese text segmentation* tool (Junyi, 2020).

Another method of statistical analysis is the vectorial semantics approach of Neuman and Cohen (2014). They argue that the attributes commonly utilized in other NLP tasks for personality profiling, specifically LIWC and n-grams, lack the flexibility to maintain accuracy across diverse scenarios. For instance, a person might behave and write differently in a school setting and a private setting. With this in mind, Neuman et al. (2015) propose vectorial semantics as an alternative to traditional machine learning to detect lone wolf perpetrators. Vectorial semantics operates under the assumption that a word has a close relationship with the words frequently co-occurring with it. This assumption paves the way for representing a word as an n-dimensional vector, with the dimensions being defined by the count of co-occurring words chosen. The direction and magnitude of the vector are then determined by the frequency of each co-occurring word's appearance in tandem with the word being modeled. The words modeled can be chosen at will from any corpus of text, making vectorial semantics more flexible than other commonly used alternatives. In their work, they propose 13 different vectors based on previous research.

The aforementioned vectorial semantics method is an example of an open vocabulary approach to NLP. Arnoux et al. (2017) use an open-vocab approach utilizing the popular GloVe framework, introduced in Subsection 2.2.2. The GloVe framework uses the same steps as the vectorial semantics presented above, requiring a pass over the corpus to be used to learn word co-occurrences. When employing GloVe vectors as features to a Gaussian Processes (GP) model, the authors reported an average increase of 33% in accuracy over previous top-tier results in personality prediction, all while using only one-eighth the amount of data. They further explained that the GloVe features and GP model equally contribute to the performance, which means that GloVe embeddings may increase the performance over other open-vocab approaches when applied to the same ML models.

User Specific Features

User-specific features are characteristics unique to a user profile on a social media platform and might help provide further insight into a user's lifestyle and personality. The availability of these features varies depending on the specific platform. For instance, the

myPersonality dataset provided researchers with data on the user's number of friends, posts, and likes. The same type of data can be extracted from Twitter with the number of followers, amount of likes, and publicly available posts.

Incorporating user-specific features has proven advantageous in enhancing the accuracy of personality prediction tasks. Plank and Hovy (2015) found that adding user-specific data such as gender and the number of followers improved accuracy when predicting personality traits. Further corroborating this, the work of Markovikj et al. (2013) shows that there exists a significant correlation between user-specific features and personality traits. Among the Big 5 personality model's dimensions, *extroversion* showed the strongest correlation with user-specific features. In particular, this trait was tightly linked to the number of friends a user has and the groups they are part of. These findings underscore the value of user-specific features in predicting personality traits.

3.4 Models Used

The choice of a machine learning model can often determine the success of extracting latent information from the given features. In previous work, a wide variety of machine learning models have been utilized. As mentioned in Section 3.2, Tadesse et al. (2018) used an XGBoost model on labeled samples from the myPersonality dataset, outperforming the baseline models used, specifically being Support Vector Machine (SVM), logistic regression, and gradient boosting. Using the same dataset, Tandera et al. (2017) sought to find the best-performing model, comparing various traditional and newer machine learning models. Traditional models included Naïve Bayes, SVM, logistic regression, gradient boosting, and Linear Discriminant Analysis (LDA), all of which were tested using a 10-fold cross-validation approach. On the other hand, the newer models used for comparison were all types of neural networks, specifically Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and a one-dimensional Convolutional Neural Network (CNN). Their study concluded that the deep learning networks, particularly the LSTM linked with the 1D CNN architecture, yielded the best results. Kaati et al. (2016) used an Adaboost model with classification trees on features from LIWC and undisclosed topic models. Neuman et al. (2015) opted for statistical models, specifically a binary logistic regression model, a tree classification with Chi-squared Automatic Interaction Detection (CHAID) and 10-fold cross-validation, and K-Nearest Neighbors with the same cross-validation setup. This broad array of studies indicates the exploration of both traditional and modern deep learning techniques in prior research.

4. Data

To build a model for predicting school shooters based entirely on the posts they publish, it is essential to have both relevant and sufficient data, although difficult to acquire. This chapter presents the data used in the thesis and provides some insight into the reasoning behind the choice of each dataset. The steps taken to preprocess and prepare the data for the prediction pipeline are also described.

4.1 School Shooter Datasets

During the preliminary project leading up to this thesis, the sparsity of texts written by school shooters became clear. Although heavily reported on by news outlets due to the particular saliency of their crimes, the number of school shooters is low compared to other types of criminals. Coupling this with the fact that only a subset of school shooters leave behind written documents or social media posts makes creating a dataset of a satisfactory size difficult. The original intention of this thesis was to collect data from school shooters' accounts on social media such as Twitter and Facebook and use it to make predictions. However, as moderation on these platforms is getting increasingly strict and effective, most, if not all, social media accounts linked to persons suspected of committing school shootings are quickly taken off their respective social media platforms. Therefore, the data collected on school shooters is sourced from posts and texts salvaged before the shooters' accounts were banned or texts made public. The following subsection will detail the collection of said data.

4.1.1 "Shooters' words" by Peter Langman

As the amount of accounts linked to known mass shooters still open for public eyes is very limited, large portions of the data collection process have relied on the work of Peter Langman, Ph.D., and his online database of known American school shooters, introduced in Subsection 3.2.2. Seeing as this database has been used by similar studies (Neuman et al., 2015; Kaati et al., 2016; Neuman et al., 2020) and is procured by an expert in the field of school shooter psychology, this was deemed as a reliable source of documents. With permission from the creator, the documents from the website section called "Shooters' words" have been collected. This section contains both original texts from the perpetrators and a collection of social media posts from shooters active on social media. The texts were either presented as PDF files or images. To avoid any noise from artifacts from PDF extraction software, these files were manually scraped. Where picture quality allowed it, each image was transcribed to text documents. Sentences and words that were difficult to make out were ignored to prevent any contamination of the original documents' textual data. The collected dataset consists of 973 texts from 25 school shooters.

4.1.2 The Twitter Archive of a Mass Shooter

The second source of texts written by school shooters is the archived tweets of the mass shooter Randy Stair (Internet Archive, 2020). Stair did not commit a school shooting but instead chose to open fire at his workplace. He is nevertheless considered as exhibiting similar traits to a school shooter due to his idolization of previous school shooters and numerous mentions of school shootings in his posts. Below are two excerpts from a journal kept by Stair before the shooting. We believe that the texts written by Stair leading up to his attacks are similar enough in nature to the texts collected from the previously mentioned school shooter dataset to warrant adding them to our dataset.

> Little does he know that I'm going to forever ruin "Anymore" for SR by using it as the center piece of a high school massacre

> As each day passes I feel less and less welcome on planet Earth. I wish I could go out and shoot up a school so bad, like my college campus. There'd be no way to kill enough people though . . . . I want to kill thousands . . . . . not just three to twelve..

Stair was particularly active on Twitter, amassing 31545 tweets across his accounts. Many of these were deemed irrelevant to our dataset and removed. A rough cut was made based on the date a tweet was posted to prevent being too specific about what tweets to include. References to violence and other school shooters can be seen to become more prevalent as the years go on. The date for the split was therefore chosen to be 13.08.2016, as we argue this is a time when Stair is exhibiting an increased interest in violence and revenge. After this cut, the final portion of tweets used is 2051 out of the original 31545.

4.2 Non-Shooter Datasets

Three datasets were chosen to represent the majority class, that is, people that are not school shooters. As the original plan of the thesis was to use techniques within the field of personality prediction to screen for school shooters, many of the datasets previously presented in Section 3.2 pertain to this area of research. Although these datasets are intended for use in personality prediction, they offer a high standard as structured datasets with exclusively English content from different parts of the web. They were therefore viewed as suitable for the purposes of this thesis. The school shooter datasets span a range of different formats, varying from Twitter posts to manifestos. The choice of datasets to represent the majority class, therefore, weighted characteristics that could match those of the school shooting datasets. Three datasets were chosen as especially suitable for the task. These were the UMass Global English on Twitter, myPersonality, and Stream of Consciousness datasets. Details of each of these three are presented below.

4.2.1 UMass Global English on Twitter

The UMass Global English on Twitter dataset is a dataset randomly sampled from publicly available tweets from over 130 countries. The University of Massachusetts Amherst collected and made the data available in 2016. The dataset was chosen due to its representation of *social media English*, which can differ drastically from English written in

academic or public settings. Each post is annotated with one of six labels ranging from "definitely english" to "definitely not english". To ensure that only tweets that were written in English were used, only tweets labeled as "definitely english" were included in the final dataset. The final amount of tweets used from the UMass dataset is 5086.

4.2.2 MyPersonality

As mentioned in Subsection 3.2.1, myPersonality was found to be of special interest for the purposes of this thesis due to its representation of English on social media sites. The myPersonality dataset was collected by the Facebook app using the same name. Users of the Facebook app were asked to take a short questionnaire of psychological questions to give the users a quick overview of their psychological profile. Users could then choose to share their social media data and posts with the application. In total, 6 million users took the questionnaire, and 40% of them opted to share their Facebook data. The final dataset contains anonymized Facebook profiles and their posts labeled with scores for each of the big five personality traits. MyPersonality has since been discontinued, and its original creators are no longer distributing the dataset. During a workshop on personality prediction by Celli et al. (2013)[1], a subset of the original dataset was made public. Our thesis utilizes this version of the myPersonality dataset. In total, this subset contains 9917 of the original posts. We chose to include all these in our final dataset. An example of a post is shown below to illustrate the type of language used in social media settings.

> HOW DID WE MEET???? Everyone play this game! Copy and paste this phrase on your profile, you will find it amusing to remember how you met and how you know each of your friends! Before you do that answer for me. Thank you and I can't wait to hear from you!!!!!

4.2.3 Stream of Consciousness

Another popular dataset is the Stream of Consciousness dataset by Pennebaker and King (1999), containing 2467 student essays. The dataset comprises informal essays written by students participating in a two-week summer school course on health psychology between 1993 and 1996. The students were asked to write an essay each day for at least 20 minutes and write about whatever came to mind when presented with a topic. The given assignments mainly concerned each individual's state of mind or significant events in the students' lives. Given the similarity in structure between these essays and blog posts or diary entries, this dataset is arguably a suitable comparison for the texts by school shooters mentioned in Subsection 4.1.1. The sample below is a cherry-picked text from the dataset, but it still captures the overall structure of each entry in the dataset.

> I wish people would come see me. I get kind of lonely sometimes or maybe it is that I don't want to be doing my work right now. I kind of like homework though because when I am done I get a feeling of accomplishment. That is also why I like to run. Because when I am done I get a feeling of accomplishment. I am really nervous about my English paper. I don't think it is very good. I am going to rewrite it now since I gave been rambling for twenty minutes. I need to pray. I wonder if God is always listening. Oh well, I'll stop now.

[1] https://www.kaggle.com/competitions/twitter-personality-prediction/overview

Each essay is written individually and has an average length of 653 words. The dataset includes binary annotations for each of the Big-5 personality traits, indicating whether or not the student exhibits a particular personality trait, but these annotations were not utilized in this thesis.

4.3 Data Preprocessing

For our preprocessing step, all data from the previously mentioned datasets were combined into a single data frame. This aggregated data was then passed through a preprocessing pipeline. The specific steps involved in this pipeline are presented below.

4.3.1 Partitioning the Data

To ensure uniformity of input data for each model in this thesis, we tokenized each post, splitting it into individual tokens separated by spaces. The posts were then split or truncated based on the token count. Our literature study identified that the most used sequence lengths for NLP tasks are around 512, though some studies use a shorter length of 256. It was decided that both approaches should be examined. Hence, posts exceeding the specified limit (512 or 256) were divided into multiple posts. For instance, a post with 1200 tokens would be segmented at the 512th and 1024th tokens, and so on, until fewer than 512 tokens remained. If a post segment resulting from this split contained fewer tokens than a predetermined minimum, it was discarded, effectively truncating the trailing tokens. We could not experimentally confirm a best-suited truncation limit and therefore chose an arbitrary minimum of 20 tokens for a truncated text. This approach was applied for both a maximum length of 512 and 256 tokens per post.

4.3.2 Text Cleaning and Preprocessing

Each post was then sent through a basic preprocessing pipeline. If a post ended up as a 0-length string after the preprocessing pipeline, it was removed from the dataset. The following preprocessing steps were applied to all posts:

1. **Removal of XML and HTML artifacts:** Some posts were retrieved from documents containing XML and HTML tags. These were removed by passing each post through the *BeautifulSoup*[2] library's *get_text()* method.

2. **Tokenization:** Each post was then tokenized with NLTK's regular expression tokenizer with the RegEx *[\w']+*, removing punctuation and splitting on spaces. Later, a choice to also use NLTK's *word_tokenize()* method to preserve punctuation was made. The tokenizers were used individually and exclusively from one another, resulting in two differently tokenized versions of the datasets. The motivation behind this was that the word embedding model FastText utilizes a lot of sub-word information gathered from punctuation and strings like *'s* or *'t*.

3. **Lowercasing**: Transform all tokens to their lowercase form.

[2] https://pypi.org/project/beautifulsoup4

4. **RegEx cleaning:** Lastly, each post was processed by a custom RegEx pipeline. This pipeline consisted of four steps:
    - Remove URLs and hyperlinks.
    - Remove Twitter user handles by matching on "@" mentions.
    - Remove hashtags, but keep the words in the hashtag.
    - Normalize words and punctuation to limit repetition, preventing words like "yeaaaa" or "okayyy...". Letters repeating more than two times were truncated to allow only two consecutive occurrences of the same character. Similarly, repeated punctuation marks were normalized to a single occurrence.

4.4 Dataset Statistics

4.4.1 Length for Cutoff and Truncation

Determining the maximum length for a specific text sequence requires considering the dataset being used and the task at hand. Ideally, we want a length that minimizes both truncation and padding, though a balance often needs to be struck between these two factors. The decision could be based on the median length or the average length of the dataset. In this thesis, we opted to use the average length of each post as our cutoff point. Table 4.1 shows the average and median length of each post, both before and after preprocessing and splitting.

Split	Average Length	Median Length
-	556.96	80.0
512	302.15	56.0
256	259.04	70.0

Table 4.1: Average and median length based on split point

Based on the preliminary literature review, a max length of 512 was initially chosen. After another review of the dataset, a shorter max length of 256 was chosen to reduce padding further. Looking at the median length of the dataset, one could argue that the max length parameter could be set even lower. The choice of cutoff threshold will be addressed in more detail in Chapter 7.

4.4.2 Word and Class Distributions

A word cloud was constructed for both classes to get an overview of the language used by school shooters compared to non-shooters. These were constructed after preprocessing the datasets. However, splitting was not performed. After applying the previously described preprocessing pipeline to each post, lemmatization was applied to normalize the text. The resulting word clouds are presented in Figure 4.1 and Figure 4.2.

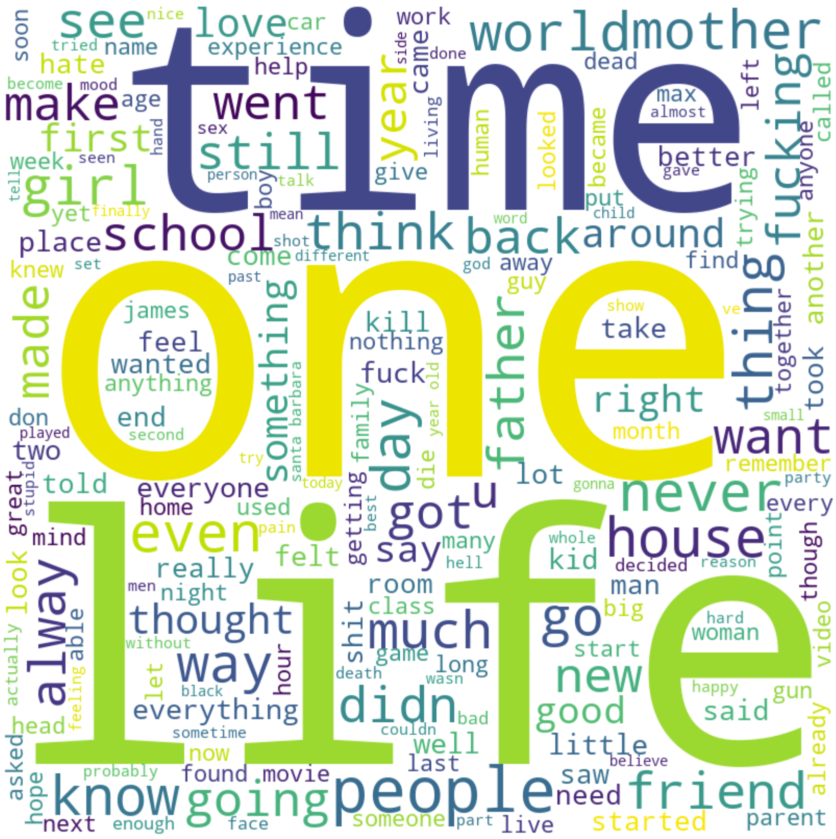


Figure 4.1: School shooter wordcloud

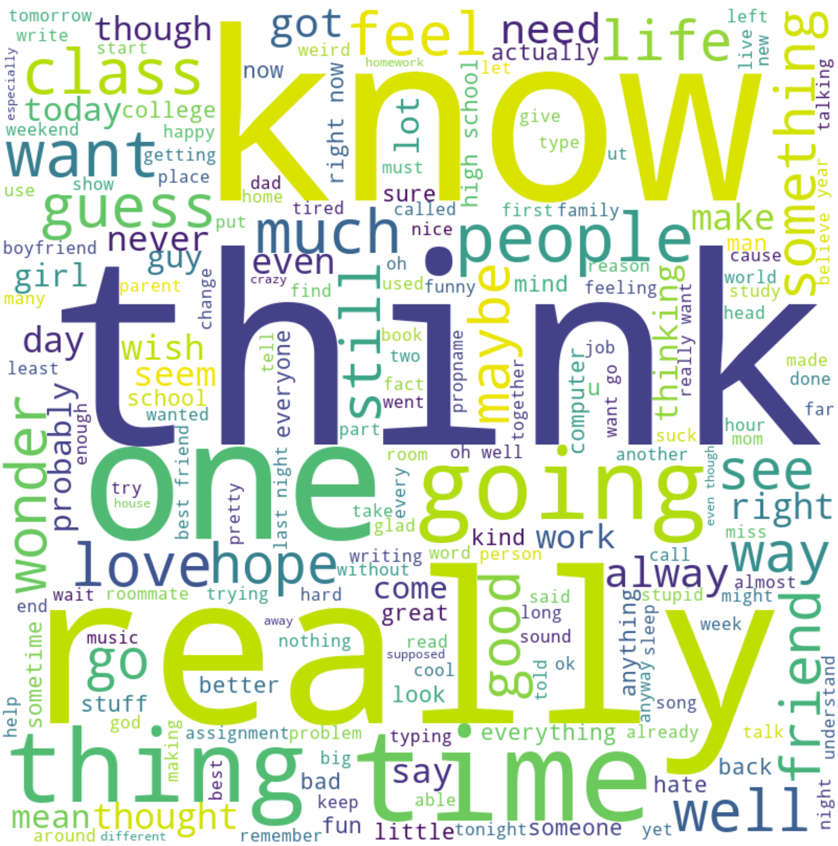
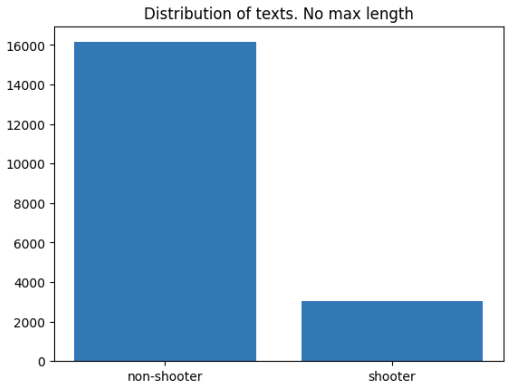
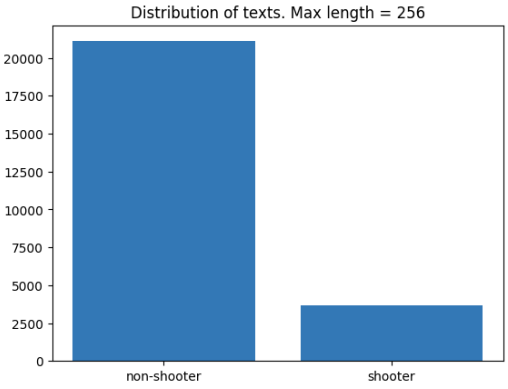


Figure 4.2: Non-shooter wordcloud

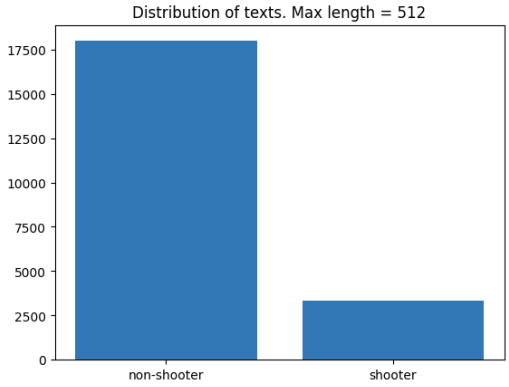
As anticipated, after preprocessing and splitting the data, there was an increase in the total number of texts from both school shooters and non-shooters. However, the ratio between the two classes remained consistent, as illustrated in Figure 4.3. Figure 4.3a depicts the original distribution, while Figure 4.3b and Figure 4.3c (in the bottom portion) display the text distributions after applying the two aforementioned maximum length splits. The final ratios were approximately 0.19, 0.17, and 0.18, respectively.



(a) Text distribution without max length. The ratio is approximately 0.19



(b) Text distribution with a maximum of 256 words. The ratio is approximately 0.17



(c) Texts distribution with a maximum of 512 words. The ratio is approximately 0.18

Figure 4.3: Class distribution based on max length cutoff point

5. Architecture

Following the collection of the aforementioned datasets, the next step involves extracting meaningful features and constructing models to utilize them. This chapter provides an overall outline of the architecture, including features, models, evaluation, and the final solution. However, the specifics of these components and their implementation, which form a substantial part of the experimentation in this thesis, will be discussed in detail in Chapter 6. The aim of this chapter is to lay the groundwork for understanding the thesis architecture and overall pipeline, which is depicted in Figure 5.1.

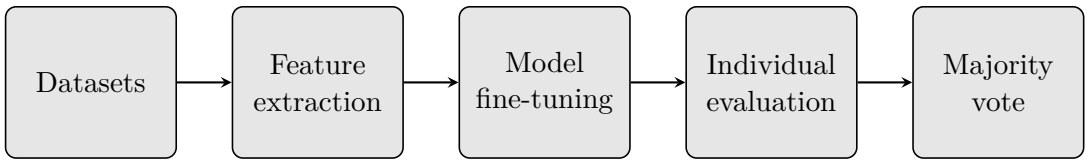


Figure 5.1: Diagram of the thesis pipeline flow

5.1 Features and Feature Extraction

A machine learning model's performance relies on high-quality data to perform well. Introducing noise in the data or just poor-quality data can drastically impact model performance during training and inference. Hence, a set of high-quality data is required to attain the best possible model performance. This section details the set of features selected for use in the experiments conducted in this thesis.

5.1.1 TF-IDF

Based on the results of previous tasks on text classification (Ramezani et al., 2018), the Term Frequency-Inverse Document Frequency (TF-IDF) approach to text representation still holds up surprisingly well on certain NLP tasks. Despite its simple nature, it was deemed a relevant feature to test. The TF-IDF features were employed in the same manner as LIWC features to establish a baseline for comparison with the more complex deep neural networks and Large Language Model (LLM) approaches.

5.1.2 LIWC

As outlined in Table 3.2, a large portion of our selected primary studies have generated feature sets through the Linguistic Inquiry and Word Count (LIWC) framework. Con-

sequently, this framework is similarly utilized in this thesis, with the specific use of four different versions: the 2001, 2007, 2015, and 2022 dictionaries. Feature extraction was performed by utilizing the licensed LIWC software. LIWC features, defined by important categories for psychological and linguistic analysis (Pennebaker et al., 2001), have been widely used in previous works on automatic personality prediction and text analysis. They often form a baseline predictor for comparison with other experiments, as seen in Kaati et al. (2016) and Tandera et al. (2017).

5.1.3 Word Embeddings

Numerous studies on personality prediction and text analysis have had great results utilizing word embeddings as their features. Further, in the specific domain of detecting school shooters, Neuman et al. (2020) employed a simple word embedding scheme to create word vectors from a corpus of school shooter texts. Aside from this basic approach, more sophisticated word embedding schemes exist, each with unique advantages. For the purpose of this thesis, we performed our experiments utilizing three different word embedding schemes that have shown good results in previous work: GloVe, FastText, and BERT embeddings.

In order to rule out data variation as a cause for performance differences, all embedding schemes were fed the same texts before preprocessing. Additionally, to accommodate for each word embedding scheme's strengths, the input data was sent through their respective preprocessing pipeline before being converted into word embeddings.

GloVe Embeddings

GloVe embeddings were extracted using the PyTorch *torchtext.vocab*[1] package, containing pretrained word vectors. The word vectors used were the *Wikipedia 2014 + Gigaword 5*[2] and the *Common Crawl 840B*[2] sets. The first set of embeddings is available in dimensions of 50, 100, 200, and 300, while the latter set is exclusively presented in the 300-dimensional form. Two different dimensionalities, 50 and 300 specifically, were chosen to study their effects on prediction accuracy. The pretrained embeddings were used in place of a dedicated embedding layer from an MLP. Since the dataset used in this thesis is relatively small in comparison to the dataset used to train the pretrained embeddings, it was decided to keep the embeddings frozen to avoid word embeddings being updated on the basis of outliers that may occur in such a limited dataset. This also holds true for the additional embedding schemes used.

FastText Embeddings

FastText embeddings were employed to see if subword information could help improve performance. Extracted using the same package as for GloVe embeddings (*torchtext.vocab*[1]), the specific FastText variant used were the *wiki en vec*[3] English pretrained word embedding model. These embeddings are only available as 300-dimensional word vectors.

[1] https://pytorch.org/text/stable/vocab.html
[2] https://nlp.stanford.edu/projects/glove/
[3] https://fasttext.cc/docs/en/pretrained-vectors.html

BERT Embeddings

Driven by the motivation of Large Language Models' (LLMs) impressive performance on downstream NLP tasks, we wanted to investigate the capability of the embeddings created by these models. LLM-generated embeddings aim to overcome the limitations of embedding schemes like GloVe. GloVe embeddings are static in the sense that they will be the same for a given word, regardless of the context it is placed in. In contrast, LLM-based embeddings, built on the transformer architecture, factor in the relationships between each word in a text sequence, making them more context-sensitive. Although more computationally expensive than GloVe and FastText, the performance gain these embeddings might yield should not be overlooked.

The LLM-produced embeddings used for this thesis were 768 dimensions long and generated by the BERT model *bert-base-uncased*[4], downloaded from *Hugging Face*. In accordance with standard practice when using these models, no preprocessing was done on the dataset before passing it onto BERT's tokenizer. The *bert-base-uncased* embedding model uses the *WordPiece* tokenizer originally proposed in Shuster and Nakajima (2012). The pipeline to generate BERT embeddings was constructed with the *Hugging Face's Transformers* library, utilizing the following modules:

- **AutoTokenizer** to download and use the pretrained tokenizer for BERT.
- **AutoModel** to download and use the pretrained BERT model *bert-base-uncased*.
- **pipeline** to create a transformer pipeline. This pipeline takes a text sequence as input, tokenizes it, and passes it to the *bert-base-uncased* model. The output is a sequence of 768-dimensional word embeddings.

Overview of Applied Features

Table 5.1 details the tested combinations of feature sets and models. A selection of classical models was tested using LIWC and TF-IDF features, while different word embedding schemes were tested with all models. Moreover, the combined LIWC and word embeddings feature set was exclusively assessed with deep learning-based classifiers. The models, in addition to their implementation, are detailed in the following section.

Model	TF-IDF	LIWC	LIWC + Embeddings	Word Embeddings
nb	x	x		x
svm	x	x		x
knn		x		x
xgboost		x		x
gaussian		x		x
cnn			x	x
bilstm			x	x
bert				x
roberta				x
distilbert				x
albert				x

Table 5.1: Overview of which features are tested on which models

[4] https://huggingface.co/bert-base-uncased

5.2 Classification Models

The classification models used in this thesis were all chosen based on their performance presented in previous work on NLP tasks. As the field of NLP has progressed, deep learning approaches to NLP tasks have become increasingly common. The latest advancement in text classification is LLMs with state-of-the-art performance on nearly all standard NLP benchmarks. We have therefore chosen to employ a collection of models from each of these approaches to evaluate their performance for our specific use case. Traditional machine learning models like SVM and Naïve Bayes are employed as baseline classifiers. Deep learning architectures are then tested to see if hidden relationships between words can be found via word embeddings. Ultimately, various transformer architectures were utilized to test the performance of Large Language Models (LLMs) on our classification task. A summary of the model architectures and frameworks used is shown below:

- **Naïve Bayes** sklearn.naive_bayes.GaussianNB
- **SVM** sklearn.svm.SVC
- **KNN** sklearn.neighbors.KNeighborsClassifier
- **XGBoost** xgboost.XGBClassifier
- **GP** sklearn.gaussian_process.GaussianProcessClassifier
- **CNN** torch.nn.Conv1d
- **biLSTM** torch.nn.LSTM
- **BERT** transformers.AutoModelForSequenceClassification("bert-base-uncased")
- **RoBERTa** transformers.AutoModelForSequenceClassification("roberta-base")
- **DistilBERT** transformers.AutoModelForSequenceClassification("distilbert-base-uncased")
- **ALBERT** transformers.AutoModelForSequenceClassification("albert-base-v2")

In order to optimize performance, a hyperparameter search was conducted for all the listed models, aiming to identify the most suitable configurations for the given task. The final architectures and hyperparameters are detailed in their respective experiments in Chapter 6. As both the *scikit-learn* models and the pretrained models fetched from the *Transformers* library have kept a default architecture besides the hyperparameter tuning, these will not be described in detail as this is not our work.

Neural Network Architectures

Classical machine learning models are relatively simple to perform a forward pass through. Deep learning models, however, have several layers and dimensions that can be modified for different results. A brief description of the architecture and a forward pass for the CNN and biLSTM models will be presented below.

CNN Architecture

Executing a forward pass through a Convolutional Neural Network can be visualized as shown in Algorithm 5.1. Dissecting the algorithm, the neural network is assembled with several distinct layers. First, it consists of three one-dimensional convolution layers, each with its own filter sizes and counts, with in-channel sizes tied to the embedding dimension. Second, the high-dimensional output from these convolutional layers is reshaped or 'flattened' into a one-dimensional vector. This transformation is necessary to connect the multi-dimensional outputs from the convolution layers to the subsequent fully connected layer. The fully connected layer comes into play next, processing the flattened input from the convolutional layers' output. This layer's function is to utilize the high-level features extracted by the preceding layers for final predictions. Lastly, a single dropout layer with a modifiable dropout rate is introduced. This layer serves an essential role in mitigating overfitting by randomly nullifying a fraction of input units during the training process.

Algorithm 5.1 Convolutional Neural Network

Require: Input data x, Convolution list *conv_list*, Kernel size *kernel_sz*
Ensure: Output result *out*

```
x_conv_list ← []
for each conv_layer in conv_list do
    conv ← conv_layer(x)
    conv ← relu(conv)
    x_conv_list.append(conv)
end for
x_pool_list ← []
for each x_conv in x_conv_list do
    x_pool ← max_pool1d(x_conv, kernel_sz)
    x_pool_list.append(x_pool)
end for
x_flattened ← concat(x_pool_list)
out ← fully_connected(x_flattened)
out ← dropout(out)
out ← sigmoid(out)
return out
```

BiLSTM Architecture

While the biLSTM architecture presented in Algorithm 5.2 may appear substantially different from the preceding CNN architecture, the underlying layer structure remains relatively unchanged. Initially, a variable number of Bidirectional Long Short-Term Memory (biLSTM) layers are utilized, each with distinct input and hidden sizes. The input size corresponds to the dimensionality of the tested embedding, while the hidden size can be modified using a configurable hyperparameter. Following this, the final layers mirror those of the CNN architecture, encompassing a dropout layer featuring a customizable dropout threshold, a single fully connected layer that receives input from the final hidden states of the biLSTM layers, and ultimately, a sigmoid function to generate the output prediction. Consequently, a forward pass through the network would adhere to the sequential process depicted in Algorithm 5.2, presented on the next page.

Algorithm 5.2 BiLSTM and Fully Connected Network

Require: Input data x
Ensure: Output result *out*
1: $hidden\_states \leftarrow \text{biLSTM}(x)$
2: $state\_forward \leftarrow \text{get\_hidden\_state}(last\_hidden)$
3: $out\_backward \leftarrow \text{get\_hidden\_state}(first\_hidden)$
4: $out\_flat \leftarrow \text{concat}(state\_forward, out\_backward)$
5: $out \leftarrow \text{dropout}(out\_flat)$
6: $out \leftarrow \text{fully\_connected}(out)$
7: $out \leftarrow \text{sigmoid}(out)$
8: **return** *out*

5.3 Evaluation of Model Performance

To ensure a fair comparison for each model, a common method of evaluating performance had to be set. It's important to note, however, that the most appropriate metrics for evaluation will vary based on the task at hand. From Subsection 4.4.1, we saw that the dataset is heavily skewed towards non-shooters with an approximate 4/1 ratio. Due to the minority class being so underrepresented, a training step using accuracy as a performance metric would achieve an 80% score merely by classifying all texts as the majority class, resulting in no correct classification of school shooters. A more appropriate strategy would involve shifting focus to metrics that emphasize the correct classification of both the majority and minority classes, which prompts the use of the F-score. Nevertheless, due to the profound severity of actions committed by school shooters, it was deemed critical to detect most, if not all, posts authored by school shooters, even if it leads to misclassifying some instances of the majority class. After all, the premise of this thesis has been centered around the inclusion of human controllers at the end of the loop, who possess the capability to identify and rectify any misclassifications. This supports the use of recall, which does not assign equal importance to the minority and majority classes but instead prioritizes accurate classification of the minority class. As a result, the final decision favored the F_2-score, which balances precision and recall, with a slight tilt towards recall. All experiment results, however, are presented using the metrics F_1-score, F_2-score, precision, and recall, to facilitate their comparison. For a more detailed discussion on the choice of evaluation metric and its limitations, see Subsection 7.1.4.

5.4 Final Majority Vote Solution

The final experiment of this thesis attempts to improve results further by utilizing a majority vote classifier and marks the final part of the full architecture. This solution aims to take the best-performing models from all of the experiments. Each model's classification vote is weighted according to its performance on the test set. This implies that votes from lower-performing models will carry less weight than those from higher-performing ones. This experiment will be further detailed in Subsection 6.1.6.

6. Experiments and Results

The upcoming chapter will delve into the experiments conducted and their outcomes. The chapter is divided into three sections. Firstly, a plan for each experiment will be presented along with the research questions they aim to address. Secondly, the experimental setup will be described, including the technologies used and their implementation. Finally, the results from the performed experiments will be unveiled. These will then be subject to further analysis in Chapter 7.

6.1 Experimental Plan

To ensure a structured experimental process, an experimental plan was developed. Consisting of three parts, the experiments aim to answer the first two research questions presented in Section 1.2. Each part is carried out as a series of experiments and builds upon each other. The first part describes the exploration of the most meaningful features or linguistic cues in the search for a potential school shooter. Subsequently, the second part, including experiments 2–5, aims to investigate the feasibility of utilizing written posts to evaluate an individual's likelihood of performing a school shooting, as captured by the second research question. The third and final part, constituting Experiment 6, involves implementing a combined majority vote solution to enhance prediction performance.

6.1.1 Experiment 1: Extraction and Examination of LIWC and N-Gram Features

The first experiment aims to explore which linguistic cue is the most discriminative in profiling a school shooter, which is strongly connected to answering the first research question. Two approaches will be explored, the first of which is using the TF-IDF, a unigram approach, for finding patterns in specific words utilized by school shooters vs. non-shooters. The second approach is to utilize the framework of Linguistic Inquiry and Word Count (LIWC), which was found to be useful in most of the related work on lone wolf perpetrator and school shooter detection. As this thesis also utilizes a part of the school shooter database by Langman (2022), one could compare the features found most selective to the features found by others, establishing a baseline for comparison that will be expanded upon in subsequent discussions.

6.1.2 Experiment 2: TF-IDF and LIWC Features as Input to Machine Learning Models

Building on the results of Experiment 1 and related work by Tandera et al. (2017), an experiment to investigate the predictive power of LIWC features and TF-IDF vectors was devised. Experiment 2 aims to implement these measures as features for machine learning models shown to have a good performance on previous NLP tasks like personality prediction and text classification.

6.1.3 Experiment 3: Using Word Embeddings as Features

Experiment 2 attempts to classify texts based solely on the words used in each text. However, it does not take into account the context in which each word usually occurs. Experiment 3 aims to use word embeddings to utilize their inherent contextual information based on large-scale corpus training. The word embedding schemes BERT, GloVe, and FastText, with the dimensions presented in Table 6.1, have been chosen based on their performance on previous NLP tasks. Due to their relatively high dimensionality compared to unigram and LIWC features, two deep neural network architectures were employed in addition to the aforementioned classical models. The neural networks used for Experiment 3 are a Convolutional Neural Network (CNN) model and a Bidirectional Long Short-Term Memory (biLSTM) model. This allows us to test the performance of both classical models and deep learning models on our task of text classification.

Embedding Model	**Embedding Dimension**
GloVe	50, 300
FastText	300
BERT	768

Table 6.1: Embedding models and their respective embedding dimensions

In this experiment, several aspects warrant further investigation, notably the maximum length of embeddings and the position of padding. These variables can yield different outcomes when applied to various models. Initially, we chose a standard max length of 512, a value commonly used with state-of-the-art language models. However, it was also considered advantageous to test a length of 256 since it aligns more closely with the maximum length of social media posts found on platforms like Twitter. Since a portion of the dataset contains short texts, a max length of 256 would help reduce redundant padding tokens for these entries. Regarding padding position, three options were examined. The first option is to pad at the head or the beginning of the text embedding. The second approach involves evenly distributing the padding on both sides of the text. The third and final approach is to append padding to the tail or the end of the text. Ultimately, an examination of these factors should help identify the maximum performance achievable with the chosen architectures.

6.1.4 Experiment 4: Combining Feature Sets

A common approach to improving model performance is feature combination. In this experiment, we will combine feature sets from the previous experiments and use them to train a new CNN and biLSTM classifier to see if this can improve model performance.

6.1.5 Experiment 5: Large Language Models as Classification Models

Further investigating the possibility of extracting semantic meaning and hidden relationships between linguistic features and school shooters, we wanted to test the performance of large language models on the same task. For this purpose, we will employ state-of-the-art models like BERT, ALBERT, RoBERTa, and DistilBERT. This concludes the series of individual trials exploring feature and model combinations for the purpose of accurately classifying texts written by school shooters.

6.1.6 Experiment 6: Majority Voting for Increased Prediction Performance

As a final experiment, we aim to improve upon the previous results by combining the top-performing models from each step to construct a majority vote solution. The objective is to form an ensemble of classifiers that collectively may enhance the overall prediction accuracy. In order to prevent less accurate models from skewing the results, we will assign a weight to each model's prediction based on its prior performance. Each classifier will assign a class label to an entry based on whether it identifies the entry as potentially authored by a school shooter. Depending on the assigned weight, a cumulative score will be computed. In the end, this approach could potentially lead to an improvement in the final classification performance.

6.2 Experimental Setup

This section is intended to facilitate the reproducibility of the conducted experiments. It will detail the process of feature extraction, outline the models used, and describe the specific steps undertaken during implementation.

6.2.1 Unigram and Bigram Features

Unigram and bigram features were extracted using scikit-learn's *TfidfVectorizer* class. The vectorizer was initialized with the default parameters set by the library. The extraction of unigrams and bigrams was done on the preprocessed datasets meant for GloVe embeddings. The preprocessing steps in this pipeline are described in Section 3.3.

6.2.2 LIWC Features

The extraction of LIWC features was performed with the official LIWC-22[1] tool provided by the original creators of the LIWC framework, Pennebaker et al. (2001). Feature extraction was done on both the preprocessed and non-preprocessed versions of the dataset. LIWC features were constructed for the 2001, 2007, 2015, and 2022 versions of the LIWC dictionary to allow for testing on all versions.

[1] https://www.liwc.app/

6.2.3 Word Embeddings

GloVe and FastText embeddings for this thesis were extracted using PyTorch's *torchtext* package. This package provides pretrained word embeddings for GloVe embeddings of dimensions 50 and 300, as well as pretrained embeddings for FastText vectors of dimension 300. The word embeddings were extracted after running the dataset through the preprocessing pipeline described in Section 3.3. LLM-embeddings were extracted as detailed in the preprocessing section as well.

Each word embedding type was padded in three different ways to test the effect of different padding positions on prediction accuracy. The padding schemes used in this thesis are:

- **Pre-padding**: Padding at the front of the text sequence.
- **Split padding**: Splitting the padding vector at the middle, appending half at the front of the original text sequence, and a half at the end.
- **Post-padding**: Padding at the end of the original text sequence.

The padding schemes were applied for both max lengths tested, namely 256 and 512. After computing the word embeddings, they were stored for use as pre-computed word embeddings for the experiments to come.

6.2.4 Scikit-Learn Models

For all of the models implemented through *scikit-learn*, the pipeline was the same. A grid search was applied to all models, trying every combination of hyperparameters to find the optimal combination of parameters. The *GridSearchCV*[2]-library was implemented with the following configurations:

- refit = "f2": To refit the estimator using the best found parameters for maximizing the F_2-score.
- verbose = 1: To keep track of the training progress, knowing how many fits to perform. An alternative value used was a verbosity of 3, printing the score and parameters for each fold.
- scoring = custom scoring dictionary: The dictionary consisted of four scorers from *sklearn.metrics*[3], specifically the recall, precision, F_1 and F_2 scorers, for ensuring all of the scores being outputted by the function.
- cv = *StratifiedKFold(n_splits=3)*: Defines Stratified K-Fold cross-validation with three splits to be used. This method maintains an equivalent distribution of the minority and majority classes across each fold. The number of splits was chosen to balance between achieving reliable model validation results and keeping the computation time manageable, given the exhaustive grid search performed for hyperparameter tuning.

The result of the grid search, being the final parameter configurations for the scikit-learn models using LIWC and word embeddings, can be found in Section F.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[3] https://scikit-learn.org/stable/modules/model_evaluation.html#scoring

NB Implementation

For the implementation of a Naïve Bayes classifier, the *sklearn.naive_bayes.GaussianNB*[4] module was used. Scikit-learn offers the ability to implement both versions mentioned in Subsection 2.3.3, named *GaussianNB* and *MultinomialNB* specifically, each capable of outputting a class as well as the corresponding class probability. The decision on the Gaussian Naïve Bayes version was made based on the fact that all of the feature sets, being TF-IDF, LIWC, and word embeddings, are continuous variables. Moreover, since this model lacks tunable parameters, an empty grid search was conducted. This required only a single iteration of training and serves as a baseline classifier.

KNN Implementation

The K-Nearest Neighbors (KNN) model was implemented using the classifier variant by the name of *sklearn.neighbors.KNeighborsClassifier*[5]. The grid search was initialized with the following parameters:

- n_neighbors: range(1, 21)
- metric: [“euclidean”, “manhattan”, “minkowski”]

SVM Implementation

The Support Vector Machine (SVM) model was implemented using *sklearn.svm.SVC*[6]. For the grid search, the following list of parameters were used:

- C: [0.01, 0.1, 1, 10, 100]
- kernel: [“linear”, “rbf”, “sigmoid”]
- gamma: [“scale”, “auto”]

GP Implementation

The Gaussian Processes (GP) model was put into operation using the class defined as *sklearn.gaussian_process.GaussianProcessClassifier*[7]. For this model, a grid search for finding the optimal kernel was performed, while all other parameters were set to default. The tested kernels are presented below, found through experimenting with kernels on a subset of the data and the framework suggesting the use of a WhiteKernel to account for noise.

- DotProduct() + WhiteKernel()
- RBF(length_scale=1.0) + WhiteKernel()

[4] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
[5] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
[6] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[7] https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.GaussianProcessClassifier.html

XGBoost Implementation

Although in another module, the *xgboost* library provides an interface that is compatible with scikit-learn, allowing for the same code structure to be applied for training, testing, and hyperparameter tuning. The XGBoost model was implemented using the *xgboost.XGBClassifier*[8] class specifically. The grid search was initialized with the following parameters:

- n_estimators: [50, 100, 150, 200, 250, 300]
- learning_rate: [0.01, 0.05, 0.1]

6.2.5 Neural Network Hyperparameter Search

The neural network architectures used for Experiment 3 were a CNN[9] and a biLSTM[10] model from the PyTorch library. Both architectures were trained with a random hyperparameter search for each combination of embedding type and max length. Parameters presented inside brackets are individual values rather than ranges if not stated otherwise. The hyperparameter spaces used for the random searches were:

CNN

- dropout: $[0.3, 0.4, 0.5, 0.6]$
- learning_rate: log_uniform $[0.0001, 0.1]$
- batch_size: $[64, 128, 256]$

biLSTM

- dropout: $[0.3, 0.4, 0.5, 0.6]$
- learning_rate: log_uniform $[0.0001, 0.1]$
- batch_size: $[64, 128, 256]$
- hidden_size: $[64, 128, 256]$
- num_layers: $[1, 2, 3]$

6.2.6 Transformer Model Implementation and Hyperparameter Tuning

All pretrained transformer models from Hugging Face followed an identical pipeline, which included a comprehensive stage of hyperparameter tuning. This stage sought the optimal parameters by examining a wide array of defined hyperparameter combinations. To streamline and enhance this process, the *Ray Tune*[11] library, specifically designed to optimize hyperparameter tuning, was utilized.

[8] https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBClassifier
[9] https://pytorch.org/docs/stable/generated/torch.nn.Conv1d.html
[10] https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html
[11] https://docs.ray.io/en/latest/tune/index.html

Tuning Setup

The *tune.run* function from said library was initialized with the following configurations:

- num_samples = 20: The number of combinations to try. For a tradeoff between time and performance, a number of 20 combinations were selected.
- resources_per_trial={"cpu": 2, "gpu": 1}: Number of CPUs and GPUs to utilize for each trial.
- scheduler = *ASHAScheduler(metric="F2", mode="max", grace_period=3)*[12]: The scheduler to use. The scheduler bases its decision on the highest achieving model regarding the F_2-score and always runs the model for 3 epochs before applying early stopping if deemed beneficial.
- progress_reporter = *CLIReporter()*[13]: A class for reporting wanted metrics. The selected metrics were F_1, F_2, precision, recall, loss, and the values forming the confusion matrix.

Model and Hyperparameter Setup

The tuning was initialized with the same hyperparameters for all models, namely "bert-base-uncased", "roberta-base", "albert-base-v2" and "distilbert-base-uncased" as introduced in Section 5.2. In addition to the hyperparameters, we configured the model with the parameter *num_labels* = 2, which allows the download of a model version with a binary classification head. Additionally, the implementation uses *id2label* and *label2id* dictionaries to assign a "positive" or "negative" label to the respective classes. The given hyperparameter space to search within was:

- epochs = [5, 7, 10]
- train_batch_size = [32, 64]
- weight_decay = [0.0, 0.01]
- learning_rate = $[1e-3, 1e-4, 2e-5, 3e-5, 5e-5]$

6.2.7 Environmental Resources

The procedures for feature extraction and model optimization were conducted utilizing IDUN (Själander et al., 2019), a high-performance computing cluster provided by the Norwegian University of Science and Technology (NTNU). This computing cluster incorporates NVIDIA Tesla A100, P100, and V100 GPUs, which significantly enhance computational efficiency, along with a diversity of CPUs. By leveraging the SLURM workload manager[14], researchers across the university have the capability to enqueue tasks, thereby cultivating an efficient work process. This functionality proved particularly beneficial during the execution of grid searches across all models, with individual runtimes varying between several hours and multiple days.

[12] https://docs.ray.io/en/latest/tune/api/schedulers.html#asha-tune-schedulers-ashascheduler
[13] https://docs.ray.io/en/latest/tune/api/doc/ray.tune.CLIReporter.html
[14] https://slurm.schedmd.com/documentation.html

6.3 Experimental Results

This section unveils the results of each conducted experiment, providing the foundation for the evaluation and ensuing discussion in Chapter 7. Adhering to the framework outlined in Section 6.1, the following subsections present the results in order, from the first experiment to the sixth.

6.3.1 Extraction and Examination of LIWC and N-Gram Features

The first experiment was centered on identifying distinctive characteristics in the datasets that contain posts written by school shooters as opposed to those by non-shooters. Both of these datasets have been subject to feature extraction using LIWC and n-gram methods. The outcomes of each technique are presented below.

LIWC

In terms of Linguistic Inquiry and Word Count (LIWC), the primary discovery pertains to the largest positive and negative percentage disparities between the two datasets. The variances, listed in Table 6.2, show the categories that are utilized more frequently (positive differences) or less frequently (negative differences) by school shooters in contrast to non-shooters. A list of mentioned LIWC categories, along with a short explanation, can be found in Appendix E.

	Positive Differences		**Negative Differences**	
Rank	**Category**	**Difference (%)**	**Category**	**Difference (%)**
1	Clout	8.1436	WC	−19.2114
2	AllPunc	7.1808	Tone	−12.1624
3	Period	6.4156	Authentic	−5.0550
4	OtherP	3.4360	Exclam	−4.6464
5	Social	3.3532	Analytic	−4.4535
6	pronoun	2.7997	Lifestyle	−1.9745
7	socrefs	2.3312	tone_pos	−1.7535
8	ppron	2.1658	time	−1.1915
9	function	1.8928	emo_pos	−1.1764
10	you	1.5573	work	−1.1592

Table 6.2: Top 10 largest positive and negative percentage differences between shooters and non-shooters, according to LIWC category distributions

N-Grams

The next feature under consideration is n-grams. Due to the significant presence of posts from Randy Stair's various Twitter accounts in the shooter dataset, we have distinguished between a filtered collection of n-grams and a non-filtered collection for this experiment. The filtered n-grams have had their "Randy Stair specific" terms removed after extraction. Nonetheless, both the filtered and non-filtered versions are presented in Table 6.3.

Rank	Unfiltered		Filtered	
	Unigrams	**Bigrams**	**Unigrams**	**Bigrams**
1	rt	egs embersghostsquad	fucking	human race
2	egs	human race	didn	mood music
3	embersghostsquad	mood music	humans	santa barbara
4	fucking	rachael shadows	kill	father house
5	andrew	santa barbara	columbine	fucking hate
6	rachael	ghost squad	die	didn want
7	mackenzie	father house	guns	know hate
8	didn	fan art	earth	better lives
9	kill	fucking hate	soumaya	eric harris
10	die	egs wiki	video	day retribution

Table 6.3: Top 10 most frequent n-grams for shooters sorted in descending order

6.3.2 TF-IDF and LIWC Features as Input to Machine Learning Models

Classical machine learning algorithms were applied on LIWC category scores and TF-IDF-vectors to establish a performance baseline for subsequent experiments. The impact of both LIWC and TF-IDF on these shallow learning models are separately delineated in the following paragraphs.

First, a grid search was conducted on all classical baseline models across all four LIWC versions, with the setup described in Subsection 6.2.4. The top-performing architecture for each model and LIWC version, as outlined in Appendix F.1, was subsequently evaluated on the test dataset. The best results for each LIWC version are detailed in Table 6.4, whereas the exhaustive list of all results after the grid search can be found in Appendix G.1.

LIWC version	Model	Max Len.	Precision	Recall	F_1	F_2
2022	svm	256	**0.7020**	0.4786	**0.5691**	**0.5111**
2015	nb	512	0.2419	0.6357	0.3504	0.4795
2007	knn	256	0.4130	0.5147	0.4583	0.4905
2001	nb	512	0.1896	**0.7814**	0.3052	0.4811

Table 6.4: Best results for each LIWC dictionary using classical models

Following the results of the LIWC based classifiers, it was believed that the LIWC features might not be descriptive or many enough to encapsulate the information in a text sufficiently. Further experimentation was therefore done with TF-IDF-vectors for NB and SVM models. The results of this experimentation are provided in Table 6.5.

Model	Max Len.	Precision	Recall	F_1	F_2
svm	256	**0.6456**	0.7296	**0.6850**	**0.7111**
svm	512	0.6382	0.7056	0.6702	0.6910
nb	256	0.2898	**0.7856**	0.4234	0.5853
nb	512	0.2726	0.7663	0.4021	0.5625

Table 6.5: Results for SVM and NB on TF-IDF features

6.3.3 Using Word Embeddings as Features

This section presents the results of the third experiment, focusing on the use of word embeddings as features with both classical algorithms and neural networks. For the comprehensive lists of final model architectures and their corresponding test results, refer to Appendix F and Appendix G, respectively.

Classical Models

Table 6.6 describes the best model, max length, and padding with their respective performance metrics for each embedding type. Notably, all of the best performing models used a variation of the Gaussian Processes architecture, with either head or tail padding. All the best performing GP models have the same kernel setup, as shown in Table F.9.

Emb. Type	Model	Length	Pad	Precision	Recall	F_1	F_2
bert	gaussian	256	head	**0.7848**	**0.7246**	**0.7535**	**0.7359**
fasttext	gaussian	512	head	0.6817	0.5327	0.5980	0.5570
glove	gaussian	256	tail	0.6863	0.5011	0.5793	0.5297
glove_50	gaussian	256	tail	0.6464	0.3855	0.4830	0.4193

Table 6.6: Best results for each embedding type using classical models

Deep Learning Models

Emb. Type	Model	Length	Pad	Precision	Recall	F_1	F_2
bert	biLSTM	256	head	**0.7913**	**0.9074**	**0.8454**	**0.8816**
fasttext	biLSTM	512	split	0.7036	0.7814	0.7405	0.7645
glove	biLSTM	256	split	0.6064	0.8575	0.7104	0.7919
glove_50	biLSTM	256	head	0.4406	0.8733	0.5857	0.7300

Table 6.7: Best results for each embedding type using deep learning models

An identical approach to testing each combination of the models, max lengths, and padding positions was utilized for the neural networks. The best performing configuration per embedding type is illustrated in Table 6.7, along with their respective evaluation metrics on the test dataset. The best performing architectures are listed in Table 6.8.

Configuration	BERT	FastText	GloVe	GloVe-50
model	biLSTM	biLSTM	biLSTM	biLSTM
max length	256	512	256	256
padding	head	split	split	head
dropout	0.6	0.5	0.4	0.4
learning rate	0.0063	0.0079	0.008	0.0008
batch size	256	64	256	256
hidden size	64	64	128	64
num layers	2	3	3	3

Table 6.8: Best architecture for each embedding type using deep learning models

6.3.4 Combining Feature Sets

For the fourth experiment, the best architectures from Experiment 3 (Subsection 6.3.3) were tested with the best LIWC feature set found from Experiment 2 (Subsection 6.3.2). The results of applying these two models on the test dataset can be found in Table 6.9.

Model	**Precision**	**Recall**	**F_1**	**F_2**
biLSTM	0.6726	**0.9413**	0.7846	0.8717
CNN	**0.8264**	0.9029	**0.8630**	**0.8865**

Table 6.9: Results for neural networks combining LIWC 2022 and BERT-embeddings

Figure 6.1 shows the confusion matrices for neural networks using a combined feature set of embeddings and LIWC. Specifically, Figure 6.1a presents the confusion matrix for the best performing biLSTM model, while Figure 6.1b does the same for the top-performing CNN model.

		Actual	
		Positive	Negative
Predicted	Positive	417 (TP)	203 (FP)
	Negative	26 (FN)	2329 (TN)

(a) Confusion matrix for biLSTM

		Actual	
		Positive	Negative
Predicted	Positive	400 (TP)	84 (FP)
	Negative	43 (FN)	2448 (TN)

(b) Confusion matrix for CNN

Figure 6.1: Confusion matrices for biLSTM and CNN combining LIWC 2022 and BERT-embeddings

6.3.5 Large Language Models as Classification Models

Model	**Length**	**Precision**	**Recall**	**F_1**	**F_2**
albert-base-v2	256	0.8884	0.8442	0.8657	0.8527
bert-base-uncased	256	0.8791	0.9029	0.8909	0.8981
distilbert-base-uncased	256	**0.8995**	0.9097	0.9046	0.9077
roberta-base	256	0.8925	**0.9368**	**0.9141**	**0.9276**

Table 6.10: Model performance metrics for pretrained transformer models

Table 6.10 presents the best models and their respective performance metrics. For the parameters used to achieve these results, refer to Table 6.11. The full list of grid search architectures and their test results can be found in Appendix F.4 and Appendix G.4.

Parameter	**ALBERT**	**BERT**	**DistilBERT**	**RoBERTa**
max length	256	256	256	256
epochs	7	7	7	7
batch size	32	32	32	64
learning rate	3e-05	3e-05	3e-05	2e-05
weight decay	0.0	0.1	0.1	0.0

Table 6.11: Best architecture for each pretrained transformer model

6.3.6 Majority Voting for Increased Prediction Performance

The models scoring the highest in the F_2-score metric from each experiment were then assembled to make a final voting classifier. The models included are: *SVM (LIWC)*, *GP (BERT-embeddings)*, *biLSTM (BERT-embeddings)*, *CNN (BERT-embeddings)*, *ALBERT*, *BERT*, *DistilBERT* and *RoBERTa*. Each model's vote was weighted according to their respective F_2-score on the test set. The results for the new ensemble voting classifier are shown in Table 6.12.

Model	Precision	Recall	F_1	F_2
Voting Classifier	0.9652	0.9391	0.9519	0.9442

Table 6.12: Results of voting classifier using all the best models

Examining the voting classifier, some models did not positively contribute to the final performance. The SVM (LIWC), GP (BERT-embeddings) and ALBERT models were thus removed. The removal of these three models from the voting pool yielded an approximate 2 percentile increase in F_2-score. All final metrics for this classifier are shown in Table 6.13.

Model	Precision	Recall	F_1	F_2
Voting Classifier	0.9727	0.9639	0.9683	0.9656

Table 6.13: Results of final voting classifier

As a culmination of the experiments of this chapter, the final majority vote classifier was compared against the best architectures from each preceding experiment. The resulting graph is presented in Figure 6.2.

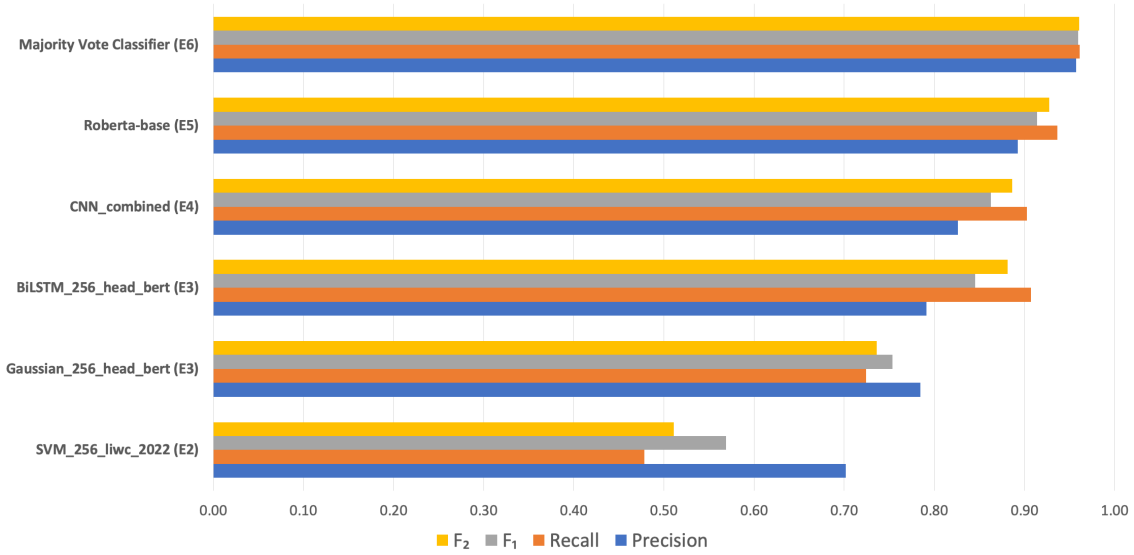


Figure 6.2: Best architectures from all experiments

7. Evaluation and Discussion

This chapter provides a thorough evaluation and discussion regarding the research carried out in this thesis. It is structured into two main sections. The first segment is an in-depth discussion about the decision-making processes that guided and shaped the experiments undertaken. Following the decisions made, the limitations and ethical dilemmas related to this thesis will be covered. In the second segment, the results derived from the experiments will be evaluated against each other as well as contextualized in comparison to past studies in the field of lone wolf perpetrator and school shooter identification. This section will try to answer the research questions and goal as described in Section 1.2.

7.1 Discussion of Experimental Setup and Planning

The upcoming section delves into crucial aspects of the research process, shedding light on the choices made that underpin this thesis. It begins with an examination of the availability of data and the data selection process before shifting focus to the steps involved in preprocessing data. Following this, there is an analysis of the feature selection process and the building and training of machine learning models. The discussion concludes with a consideration of the ethical dilemmas inherent in the application of machine learning to tasks in the field of school shooter prediction. This aims to serve as a backdrop for the forthcoming evaluation, enriching an understanding of the context and methodology driving the research.

7.1.1 Data Selection and Availability

First and foremost, it is important to note the scarcity of texts written by school shooters available at the time of writing. This may be the most significant hindrance of them all when it comes to accurately using automatic prediction as a means of detecting school shooters. As the dataset used only contains roughly 3000 texts written by school shooters, most of these being tweets, the dataset is relatively small to train a general model. Optimally, the dataset used for texts by non-shooters should have been larger as well, but this could drown out the already small minority class of school shooter texts.

Attempting to increase the size of the dataset to improve the capabilities of the models to generalize could lead to a large class imbalance. Since the minority class is so small, a tradeoff between dataset size and balance had to be made. For this thesis, three standard ways to combat imbalance were initially considered; however, it was decided not to opt for any of these and rather go for our own solution. Regular techniques such as oversampling and undersampling may negatively affect a dataset, especially if the dataset is sufficiently small. Seeing as the minority class in question, in reality, makes up a very small portion of

texts posted online, it could be beneficial to keep the minority class small in comparison to the majority class to mirror a real-world scenario. Furthermore, using an oversampling technique when the minority class is sufficiently small could cause a large portion of the resulting new minority class to have duplicate entries, making a model trained on this data perform worse on unseen instances. A suggested approach to avoid this problem is synthetic oversampling, a technique where new entries are automatically generated based on already existing data. This could work to create a more balanced distribution of the classes, although there could be a risk of increasing the noise already inherent in a dataset containing free-form written posts. Taking a look at some entries in the dataset, school shooters do not exclusively write about violence or threats in their posts. They also frequently write more mundane posts concerning what movies they like or what food they prefer. There are also cases in the dataset used where a post is written by a school shooter, thus being labeled as a school shooter text, but not containing any "incriminating" terminology or themes. An example of this is the entry presented below:

> Yesssss!!!

This post was written by a school shooter, but there is nothing that could or should be alarming about this post by itself. Therefore, one would have to take particular care when choosing what data to base the synthetic oversampling on. If the portion of labeled shooter posts available to the synthetic oversampling technique is small, there is also a risk of hurting general performance present due to less general data.

Taking this into account, the techniques commonly used to combat class imbalance were not utilized. Instead, the approach presented in Section 4.3, involving splitting of texts into portions based on a set length, was used. This method was viewed as a reasonable approach to increase the size of our dataset due to two reasons. Previous approaches for NLP often use 256 and 512 tokens as their max sequence length and justify this by showing that truncating a text after these first tokens usually do not lead to a significant decrease in models' predictive ability. The amount of tokens ultimately has to be viewed as a tradeoff between splitting texts and padding shorter ones. Some of the datasets obtained from *schoolshooters.info* are manifestos several pages long. Naturally, these entries exceed the max sequence length. Instead of being truncated, these texts were rather split into sequences of tokens equal in length to the max number of tokens. This would allow us to retain the information contained in the entire text entry and expand our dataset in the form of several max-length entries, all while preventing the creation of duplicate entries. This method could, however, introduce less informative entries due to arbitrary cutoff points as a result of the cutoff being made whenever the chosen max length is exceeded. Nevertheless, using continuous sequences of a larger text sequence has been experimented on before (Mulyar et al., 2019) and has proven to give desirable results. It was therefore decided that this method of upsampling the minority class could be a feasible approach to tackle the problem of sparse data. As seen in Figure 4.3, this did increase the amount of school shooter relevant texts, but also the number of texts written by non-shooters. The minority class remained approximately equal in proportion to the majority class in the splitting process but contained a larger amount of samples to train on.

7.1.2 Preprocessing of Data

The choice of preprocessing steps is an important part of every NLP task. Choosing the right ones for the task can affect model performance drastically. In this thesis, different

preprocessing pipelines have been tailored to varied use cases. For statistics and graphing, only punctuation removal and lemmatization were performed. This was to help the readability and interpretability of the results presented in Experiment 1. Preprocessing applied in the experiments concerning feature exploration was more thorough but still just a standard preprocessing pipeline containing the removal of stopwords, special words, and punctuation. The data was also cleaned due to some portions originating from social media and, therefore, possibly containing hashtags, user handles, and links. It was not desirable to perform lemmatization or stemming due to the possibility of losing context. Notably, this preprocessing was not applied to the data sent to the language models. This was to comply with the guidelines for the usage of these models, which state that important subword information can be gleaned from punctuation patterns and word contractions. Therefore, only raw text data was passed to these models and their tokenizers.

Initially, the same preprocessing steps were applied before feeding the text to both GloVe and FastText. However, FastText also promises the ability to learn subword information and out-of-vocabulary words. It was therefore seen as beneficial to keep punctuation before generating pretrained embeddings for each token when using this embedding model. GloVe also has pretrained word embeddings for punctuation, meaning that it should be able to handle punctuation as well. FastText, unfortunately, does not have any documented standard of preprocessing before passing input to the model, which meant that the preprocessing applied for this model ultimately came down to our own judgment. Initial trials were carried out on both the preprocessing applied to GloVe and the currently used preprocessing pipeline for FastText. The results gave inconclusive answers of which was best, as the difference in performance was only a few percentiles apart. With such a small difference in performance, this variance could well be the result of randomly initialized weights. Nevertheless, it was decided to keep punctuation for FastText embeddings due to there being no apparent downsides of keeping it.

7.1.3 Feature Selection

Several studies have looked at automatic personality prediction in order to attempt the identification of a specific group of people or for text classification. The de-facto personality model to use for such a task seems to be the Big Five personality model, supported by the multitude of papers found in the preliminary literature review. A report published by the U.S. Secret Service (National Threat Assessment Center, 2019) states that school shooters almost always had motives such as a previous grievance with a previous classmate, a wish for fame, or a desire to kill. It is also stated that most attackers had experienced psychological or behavioral symptoms. This could manifest as depressive symptoms or misconduct. A personality prediction based model was therefore seen as a good fit for this thesis. However, after discovering the lack of data on school shooter texts overall and also the total lack of such datasets annotated with personality scores, this was decided against. One could argue that it is possible to extract these personality scores using machine learning models instead. There is also the possibility of taking the job of annotating personality scores into our own hands. These two alternatives were ultimately ruled out as well. The self-annotation scheme was ruled out simply because we do not have the domain expertise to make such classifications. Automatic personality detection was decided against because this could be a cause of error that could contaminate the small amount of data available. Therefore, it was decided not to utilize personality scores as features and rather keep the problem as a simple binary classification problem. This allowed us to be certain that the texts presented as written by school shooters actually were.

Motivated by previous work on the automatic detection of school shooters, two feature sets were initially examined, namely features extracted from LIWC in addition to word vectors. LIWC features were seen as especially relevant due to the aforementioned report by the U.S. Secret Service. Kaati et al. (2016) uses LIWC features to try to identify lone wolf perpetrators from written communication. This would make the classification process more dependent on psychological markers found in the written material. Due to this, LIWC features are often seen used in work on personality prediction, such as Lima and de Castro (2014) and Tadesse et al. (2018). Kaati et al. (2016) showed promising results with the use of LIWC features as predictors of a lone wolf perpetrator. The datasets used include documents from the schoolshooters.info database; however, the texts used in their study are not exclusively written by school shooters. An important difference between the data of Kaati et al. (2016) and ours is that their texts seem to be exclusively long-form texts more geared towards ideologies. From an examination of this thesis' school shooter dataset, this is not very common at all. Only a few select texts have any mention of ideologies, meaning that common terms relating to any ideology are hard to find. This may explain the difference in results between our implementation of LIWC features as predictors versus the approach shown in Kaati et al. (2016).

The second set of features investigated in this thesis was word vectors. Motivated by Neuman et al. (2015), an initial approach was to use TF-IDF word vectors to calculate unigram representations of the corpus. This technique proved to be more accurate than LIWC features on our dataset. This could be related to another observation presented in the paper by the National Threat Assessment Center (2019), which states that there is no common profile of a school shooter. This could mean that finding common psychological features of school shooters could prove difficult despite the fact that several have common grievances and behavior. Another shortcoming of using LIWC features as predictors could lie in the way the features themselves are calculated. Both feature sets are scaled according to the size of the documents examined; LIWC is scaled by using percentage metrics, while TF-IDF is scaled according to the number of words in the document. An important difference is that each word's importance in the TF-IDF model is based on how many documents the word appears in, meaning that a word that occurs in a lot of documents is viewed as less discriminative and therefore gains a low score. LIWC does not take this into account, which could result in a worse separation of the shooter and non-shooter texts.

Unfortunately, there may be a risk of these vectorial semantics approaches not performing well on completely unseen data, such as new words or new ways of writing. This is due to feature sets generated with vectorial semantics being entirely dependent on the datasets they are exposed to. Meaning that if, for example, a TF-IDF model is trained on only the data contained in our datasets, the sample size would be very small compared to what would be desirable. A way to combat this could be to use pretrained word embeddings. Word embeddings are also often used to overcome the problem of capturing context in word vectors. By using pretrained word embeddings such as GloVe or BERT generated ones, relationships between words learned through billions of examples can be leveraged. Consequently, word embeddings were adopted as an additional feature set. The motivation is that the contextualization provided by these embeddings could assist the model in better detecting subtle differences between texts belonging to the minority and the majority classes. Transitioning from LIWC features to word embeddings resulted in a considerable F_2 increase, from 0.5111 to 0.7359 for the classical models, further increasing to 0.8816 with deep learning models. It is worth emphasizing that the performance of deep learning models was not measured on LIWC features alone, making this comparison somewhat unsubstantiated. Nevertheless, a considerable performance increase is seen when switching from LIWC to word embeddings using classical models.

7.1.4 Building and Selecting the Best Performing Models

The choice of models for this thesis was based on architectures identified in the preliminary literature review to perform well on related tasks. Simple models like Support Vector Machines and Naïve Bayes have managed to get impressive scores despite their relatively easy implementation. Although a lot of the work done in this thesis is based on what was extracted from the preliminary study, not all methods previously identified to perform well did so for the case of automatic detection of school shooters. A host of different models were applied to each set of features, all of which can be found in Section 5.2.

To find the best model for the tasks presented in each experiment, a hyperparameter search was performed for all relevant models. All classical models were tuned using a grid search with a set of predefined hyperparameter spaces. The larger deep learning models and LLMs were instead tuned using a random search on hyperparameter spaces. This random search could, in turn, mean that the absolute best hyperparameter configurations were not found. However, the performance gain of a grid search compared to a properly configured random search is often negligible at best (Bergstra & Bengio, 2012).

The choice of evaluation metric to base the hyperparameter search on largely depends on the task and datasets at hand. Common metrics to optimize for are accuracy and F_1-score. In the case of this thesis, the minority class is so small that a model training on the dataset and trying to maximize accuracy would achieve very high accuracy by always guessing the majority class. Another approach would be to optimize towards always correctly classifying the minority class. This could, however, lead to the opposite problem of what was just presented, where non-shooter texts are frequently classified as school shooter texts. Metrics that also emphasize the correct classification of all classes would therefore be more suited for our classification problem. Consequently, the selection was narrowed down to a variant of the generalized F_β-score, which encompasses both precision and recall. Opting for the F_2-score over the F_1-score was driven by its slightly greater emphasis on identifying all potential school shooters. However, a consequence of optimizing for the F_2-score instead of the F_1 is that misclassifications of the majority class are not punished as harshly as misclassifications of the minority class, potentially leading to more frequent misclassifications of the majority class. Considering the objective of incorporating human evaluators in the process, the tradeoff of a few misclassifications is deemed acceptable if it could lead to the discovery of a school shooter. This approach enables human evaluators to detect and rectify any errors that may occur. Ultimately, in the context of this thesis, the choice of evaluation metrics will largely depend on whether the goal is to correctly classify as many school shooter texts as possible or if the goal is to minimize misclassifications.

With this in mind, it is apparent that the models trained in this thesis all have a bias towards the minority class. The consequences of which can be seen in one of the posts classified as a school shooter text below.

> We are not who you think we are. We are Golden.

Although a bias towards the minority class is inherent in the models, a surprisingly good accuracy has been achieved for both classes, probably owing to the fact that the minority class is so small that its small size might outweigh some of the weighting schemes and optimizations applied to classify it correctly.

7.1.5 Ethical Considerations

As stated earlier, there is no consensus on a specific set of features identifying someone as a school shooter. This presents a problem when trying to automatically detect such an individual. What should the models look for if there are no known profiles to match against? What further complicates this matter is the profound effects it could have on their lives. Machine learning is not an exact science and will likely never be. Although technological advances continue to produce more and more accurate models, there is always a risk of error. The question is whether the occasional error is acceptable when handling a situation as sensitive as labeling someone as a school shooter. An easy solution would be to ignore this problem altogether. Making such a decision, however, would be very unwise as the models tested in this thesis are not tested on enough data to be certain that they are accurate enough to trust. A misclassification due to model error could be catastrophic if it is taken as fact. Therefore, the predictions made by the models in this thesis should not be taken as absolute; even our best models misclassify text entries that have no semblance of a school shooter text at all.

Another consideration to be made is the massive risks of privacy violations that an automatic screening procedure of this nature could impose. For the purposes of this thesis, only data written by known school shooters were collected and labeled as school shooter texts. These documents are already publicly available, most of which have been officially released by the police departments handling each incident. Therefore it was not viewed as necessary to perform any form of anonymization of the individuals listed as school shooters for this thesis. However, if a system similar to this is to be deployed in the real world, serious consideration should be taken before collecting data. A form of consent would have to be signed by each student at the school the systems were to be deployed, giving access to their private social media posts and maybe even internet activity. It could be argued that posts and accounts made publicly available are shared at the discretion and risk of the user. On the other hand, cases like the Cambridge Analytica scandal (Cadwalladr & Graham-Harrison, 2018; Wong, 2019) show that this truly is not the case. After granting consent to the issuer of such a system, a student would have to live with the fact that their every post is being monitored by a bot looking for warning signs and terms every hour, every day. The mental strain this could inflict on a person is also an important factor that needs to be addressed before a monitoring system can be deployed in the real world. There are already examples of social media monitoring applications that aim to do this out there. However, reports of their effectiveness make it hard to justify their existence for now. Administrators using these tools report too many false alarms on texts that obviously present no risk (O'Leary, 2022). The same articles also report instances of school shootings occurring at schools actively using such monitoring services (Faife, 2022). The problems faced here shed light on one of the main challenges with an automatic screening approach. To be able to flag posts, the system would have to see them first. If posts containing warning signs are posted on a platform the system does not have knowledge of, or if a user's profile is private, it naturally cannot be detected. A solution could be to monitor an increasingly larger portion of the student's online presence, but this leads us back to the previous question of privacy.

Ultimately we find that there is no good answer to the question of whether the use of screening tools, such as the one presented in this thesis, is substantiated. However, one thing becomes apparent; if a school decides to test such software, it should never be used without the supervision of a human expert able to ascertain if flagged material indeed shows any warning signs.

7.2 Evaluation of Results

This section will present an examination of the results obtained in Chapter 6, connecting them with the initial goals of the thesis and discussing whether these goals were met.

Research Question 1 *Which, if any, linguistic traits do school shooters have in common?*

The purpose of Research Question 1 was to identify any linguistic similarities among school shooters that could serve as a foundation for further exploration or to pinpoint specific keywords that could be targeted by potential screening software. Looking at the results of Experiment 1 (Subsection 6.3.1) show that there may indeed be commonalities in the texts written by school shooters. Results of the LIWC framework show distinct differences in the use of, among others, words concerning clout and personal pronouns. Other more subtle measures, like the use of punctuation and periods, also seem to differ significantly. On the other side of the spectrum, texts written by school shooters seem to contain fewer words pertaining to everyday subjects, leading to the LIWC dimensions for lifestyle, work, and positive emotions scoring lower for school shooters than for non-shooters.

The results of the analysis of n-grams in Experiment 1 seem to support the results shown from the LIWC analysis. In particular, looking at the filtered bigrams section reveals a majority of negatively loaded terms, with few, if any, positively loaded ones. Unfortunately, the size of each document among the texts written by school shooters varies, meaning that some school shooters may contribute more to the results of this analysis than others. This was apparent from the start and is also why the n-grams are separated between a filtered and unfiltered version. This filtering manages to overcome some of the noise created by certain shooters; however, long diaries, many posts, or manifestos still have a large influence over the final results. This can be seen in the form of unigrams like “soumaya” or simply the term “fucking hate”. This is written multiple times by Randy Stair in the collection of tweets gathered and only twice by Eric Harris. The rest of the dataset contains no instances of this bigram at all; however, it is still ranked among the top 10 most distinctive for shooters. Thus, an apparent limitation due to the small sample size used comes to light — the amount of different individuals sampled in the combined school shooter dataset is so small that an outlier may severely affect the results gained when examining the dataset. Therefore, insight gained from Subsection 6.3.1 should not be taken as absolute truth but rather as a hint of what tendencies might be present in texts written by school shooters. Nevertheless, there seem to be commonalities present between the texts written by school shooters contained in our dataset that are represented in the form of distinguishing unigrams and bigrams as well as LIWC dimensions. The question of whether or not this accurately represents the actual differences between school shooter texts and non-shooter texts on a larger scale is difficult to say for certain, and we, therefore, conclude that the findings from Experiment 1 have too little data to substantiate a claim of whether or not these differences are in fact true to real-world data.

The conclusions reached in the following experiments will all be affected by this premise, meaning that the results presented in the study should not be taken as fact but rather as a step on the way to understanding the potential use of linguistic features as predictors of school shootings.

Research Question 2 *How indicative are written records of an individual’s potential to become a school shooter?*

Judging from the conclusion to **RQ1**, care has to be taken when answering **RQ2** as

well, since these are two closely related questions. The results discussed in **RQ1** seem to show that there are linguistic cues contained in written records produced by school shooters. Following this early experiment, the remaining experiments try to use linguistic cues extracted from texts to determine whether it was written by a school shooter. A less humanly tangible feature set, word embeddings, was also extracted to help with this task. Experiment 2 (Subsection 6.3.2) suggests that using LIWC features alone give poor results in regard to a balanced performance. The features do not seem to be descriptive enough, making the classifiers lean heavily toward one type of prediction. Since the majority class is non-shooters, this takes the form of classifying most texts as non-shooters. Since the distribution of the dataset is roughly 20% school shooter texts, this approach would yield an accuracy score of nearly 80%. The poor performance of LIWC features in this thesis compared to the works of similar studies like Kaati et al. (2016) may be attributed to the fact that previous studies utilizing LIWC features often incorporate them with SNA-features. Unfortunately, most, if not all, social media profiles of the school shooters studied for this thesis were either taken down or non-existent, making this approach infeasible.

TF-IDF features were introduced in an attempt to alleviate the shortcomings of LIWC features. However, caution should be taken when examining the results of utilizing TF-IDF, as the problem outlined in **RQ1** persists; large text documents can influence the model performance and artificially boost reported accuracy. This is due to the approach used to increase the number of entries by school shooters. Splitting large manifestos or collections of posts into smaller entries may distribute terms specifically used by only one person across the entire dataset, meaning that the dataset now has a larger portion of documents containing person-specific terms. If the TF-IDF classifiers are taught through training that these terms are distinctive features of a school shooter, it will most likely classify the texts containing these terms as school shooter texts. If new posts written by non-shooters containing these terms are introduced to the dataset, they may very likely be wrongfully classified as school shooters. Regardless, the TF-IDF based classifiers performed reasonably well, with the best SVM configuration having an F_2-score of 0.7111.

Referencing Subsection 7.1.3, a performance gain was seen when substituting TF-IDF vectors with globally trained word embeddings, suggesting that the semantic information contained in word embedding schemes such as GloVe and FastText may help represent more attenuated linguistic cues which otherwise are hard to encapsulate in traditional TF-IDF vectors. The best classical model utilizing BERT-generated word embeddings saw only a slight increase in F_2-score when compared to the best classical model for TF-IDF based classifiers. Moreover, with BERT-generated word embeddings having a dimensionality of 768 per word, one text sequence of the minimum max length of 256 would yield a matrix of 196608 values. This dimensionality may be too high for classical models to properly be able to capture the underlying word embedding information. This becomes apparent when comparing the results of classical model architectures to those of deep learning-based models. The biLSTM architecture achieves an F_2-score of 0.8816, an increase of 0.1705 from the best classical approaches. As a final experiment on traditional text vectorization techniques, the LIWC feature set was combined with the best performing BERT-generated embeddings. Doing so yielded only a minuscule performance increase for the best CNN architecture while actually decreasing performance for the previously superior biLSTM implementation. The small increase in performance for the CNN model could be attributed to chance due to a different initialization of model weights when extending the feature set with LIWC features. The same could be said for the decrease in performance of the biLSTM architecture. This seems to further suggest that, at least for the texts contained in the presented dataset, LIWC features are not good estimators for machine learning models when trying to screen for school shooters.

As the final part of feature and model exploration, Experiment 5 shows promising results when using raw text data as input to transformer-based language models. This reinforces the impressive results Large Language Models have shown in NLP tasks, with all but one of the tested architectures outperforming our word embedding trained neural networks.

Combining the best performing models from each experiment to get a balanced voting classifier based on different feature sets yields very promising results. The performance of this voting classifier was obtained using high-dimensional word embeddings and large language models trained on huge datasets, meaning that the specific cues picked up by the model are difficult to interpret. Nevertheless, the final voting classifier achieved a very high F_2-score of 0.9656, a 0.04 increase from the best LLM, RoBERTa, highlighting the strength of such voting classifiers. When fed our testing dataset of 3000 unique texts, the voting classifier only misclassifies 28 texts, with most of these having predicted values very close to neutral, meaning that the classifier was conflicted as to what class the text belonged to. This equates to an accuracy of $\approx 99\%$, beating previous results on automatic prediction of school shooters (Neuman et al., 2015; Kaati et al., 2016). Judging from the performance of the final voting classifier, we would argue that there indeed seem to be significant linguistic cues present in the school shooter texts that hint at whether they were written by a school shooter.

Although the solution presented in this thesis achieves an impressive F_2-score, there is another aspect that has to be taken into consideration. Our final voting classifier implements an ensemble of four large language models and three deep learning models. Additionally, these three deep learning models utilize word embeddings generated by a large language model. Hence, the whole process of performing one full voting round is not exactly cheap. The best solution overall would, therefore, entirely depend on a cost-to-performance tradeoff. As is often the case with machine learning, higher accuracy means heavier computational costs, which may not be desirable for the purposes of performing screening on a dataset as large as social media posts. The large computational burden of this voting classifier also brings the speed of this classifier into question. Will the classifier even be fast enough to handle all posts being published every second? This is, in turn, related to the question of suitability linked to the use final solution. As this thesis should serve as more of a proof of concept and a presentation of the performance that is achievable, metrics such as scalability and computational cost were not deemed relevant for each model's success. However, if one wants to deploy the framework presented here, such considerations have to be made and could indeed mean that a worse-performing model would be preferred in place of the final, computationally heavy solution.

As a final remark on the performance of the voting classifier, one should be mindful of the fact that this classifier is constructed from a combination of the specific classifiers proven to perform the best on the test set. In a sense, this could combat the whole purpose of the test set itself. By cherry-picking the models performing best on the test set and then making them collectively vote on the same test set, one could argue that we are optimizing this voting classifier for the test set, defeating the purpose of a general test set with completely unseen data. While examining the results of Experiment 6, it was therefore deemed desirable to do a final test on a separate holdout set consisting of totally unseen school shooters. After feeding the holdout set to the final voting classifier, the classifier correctly classifies 7 out of 10 texts correctly. The problem of a small sample size presents itself here. Some lower quality samples are incorrectly classified as these posts contain only mundane quotes like "good bye" or "Tired of everything rn". Judging from the performance on this test set, it seems like the model does reasonably well on newly introduced data despite the previously mentioned concerns.

Research Question 3 *How suitable are machine learning methods for predicting the potential of someone performing a school shooting?*

The question of whether machine learning methods are suitable for the task of automatic prediction of school shooters is a two-sided question. Firstly, the machine learning model must have the capability of accurately classifying the texts, with little room for error due to the sensitive nature of the task in question. The second aspect of the problem is whether it is ethically responsible to release such software out into the wild, no matter how well it performs.

Based on the results gained through Experiments 2–6, it seems that an approach using machine learning for the automatic detection of school shooters is indeed possible. Despite the caveats discussed in Section 7.2, the final solution using a voting classifier achieves high performance, with only a few misclassifications. Therefore, the question of whether such a solution is suitable mostly comes down to the ethics of using such methods on real people. Referring to the discussion from Subsection 7.1.5, we believe that there is no good answer to this part of the question yet. We choose to err on the side of caution and say that if such a solution is to be used, it has to go through a thorough testing regime as well as it having to be used in combination with human professionals that will always have the final say in what decision to take. With that in mind, the models constructed in this thesis would, rather than being a fully automatic system automatically flagging school shooters, instead be a ranking system flagging individual posts for a human operator to examine closer. The motivation being that such a system could aid a human operator in finding relevant posts, thus decreasing the workload of the human operator. Hence, machine learning methods used to automatically predict school shooters should be used carefully in combination with a human operator. Only then would we consider the methods presented in this thesis as suitable for real-world use.

Goal *Automatic detection of possible school shooters based on linguistic cues extracted from their written work.*

The experiments conducted in this thesis seem to coincide with statements from referenced literature stating that there is no common profile of a school shooter, meaning that broad spanning features such as LIWC features may not fare well on their own. The need for more abstract features was addressed by utilizing word embeddings in combination with deep learning models to create highly accurate model architectures. As a last step, the aforementioned voting classifier was created by combining all the best performing models into one weighted majority vote solution that beats solutions presented in related work on the automatic detection of school shooters. We argue that the goal of creating an automatic machine learning model for the prediction of school shooter texts has been satisfied; however, future work should aim to expand the datasets presented in this thesis to allow for more general and robust models. In the end, there still have to be made significant considerations when employing automatic prediction models like the ones presented. **RQ3** tries to touch on the ethics and legalities of such implementations; however, we argue that this is a subject far too extensive to be exhaustively covered in the scope of this master's thesis. Thus, investigations into the ethics of automatic screening of school shooters stand as an important part of the future work needed to be done before any version of such automatic screening tools should be brought to life.

8. Conclusion and Future Work

The sections that follow intend to summarize the work of this thesis. In the first section, the findings, together with the contribution of our final solution, will be described as a conclusion of our work. Moreover, a section will be dedicated to highlighting research areas potentially serving as future work for others to investigate.

8.1 Conclusion

As online forums continue to grow, the amount of posts to each forum is seeing exponential growth. The task of finding a threat among all the non-threats becomes a needle in a haystack problem that is increasing in scope for every day that passes. Consequently, a manual process to screen for threats in the sea of posts is quickly becoming infeasible. This thesis aims to address the problem of screening for school shooting threats through the automatic detection of texts written by potential school shooters. To this end, we attempt to apply different NLP techniques to extract features from written text to differentiate between texts that are written by school shooters and those that are not. Analysis of a collected set of school shooter texts shows linguistic and psycholinguistic differences between these texts and texts written by non-shooters. Trying to utilize these differences as distinguishing traits, LIWC, TF-IDF, and word embeddings were all tested as input for a host of machine learning architectures. The most promising results were found when using BERT-generated word embeddings as input for both LLMs and deep learning-based architectures like CNN and biLSTM networks. Ultimately, a voting classifier consisting of the best-performing models from each individual experiment was constructed to attempt to weigh up for each model's individual weaknesses. This final voting classifier outperforms previous studies on the automatic detection of school shooter texts, attaining an F_2-score of 0.9656. Although a seemingly effective classifier, the ethics regarding the use of such classifiers are still unclear. For the proposed solution to be effective, it would need access to private social media and forum profiles. Due to this, in combination with the risk of misclassifications and the profound effects these could have, it is our belief that such a model is not yet suitable for use without a human expert monitoring results and flags set by software utilizing this architecture.

8.2 Future Work

In the research for this thesis, various areas warranting further exploration have been identified. The next section is committed to thoroughly detailing these areas, exploring the potential enhancements we believe could contribute to the process of automatic school shooter detection through written work.

8.2.1 Leveraging, Expanding and Creating Data Sources

The first and perhaps most self-evident proposal for future work revolves around the expansion of the data upon which the classification models are built. With regard to school shooter data specifically, there is a pressing need for the broader availability of data. While we have made significant use of the continually expanding schoolshooters.info database, in addition to the Twitter archive of Randy Stair gathered from the Internet Archive (2020), it is clear that the source of data remains insufficient in size. This suggests a compelling need for a significantly larger source of documents.

An ideal solution could be for social networking services to provide a portion of data for research, including data from blocked accounts that are currently unavailable via the tested APIs. Given the tendency for user accounts to be suspended in the wake of newsworthy incidents, often very quickly, there's a valid assumption that the service providers are aware of the reasons behind such suspensions. By assigning labels to these users indicative of the reasons for their suspension, like being flagged as potential school shooters, and subsequently sorting them into a category that approved researchers can access, one could swiftly extend the available data for the experiments in this field. Alternatively, researchers could leverage the public availability of the perpetrators' names, which typically emerge a few days after such an incident. This enables them to track down the online accounts associated with these individuals. By compiling lists of usernames and cross-referencing them with social networks or establishing links using alternative methods, the connection between the individual and their online persona could be established. This approach effectively bridges the gap between the user and the real-life person, offering new sources for data collection. Although understandable that the data should be removed from the masses based on the disturbing content it might include, this could have a significant impact if the data were to be made available for research purposes. In addition to the data being stored by social networking services but not fully utilized, further contributions like the previously mentioned work of Dr. Langman, which involves the collection, verification, and potential annotation of texts, could vastly enhance the data pool.

Another possibility of exploration involves harnessing the power of generative machine learning models to fabricate data that mirrors the behaviors and language patterns of school shooters. In light of advancements in text generation models that can convincingly replicate human language, an unexploited opportunity presents itself in the synthetic generation of new data grounded in previously written text. This method, while innovative, does come with a caveat — the possibility of overemphasizing characteristics linked to certain school shooters. This situation arises from the fact that the texts of many school shooters are not typically collected and publicly accessible. More often than not, these posts are kept confidential by the networking services used, or in cases where notes were physically written, they might not have been digitized and made publicly available. This yields a low amount of subjects to base the characteristics on and will not help in the case of creating a good-performing generative model. However, this can be mitigated by expanding the breadth of real, authentic data used in the generation process. The more diverse and extensive the data fed into these models, the better equipped they become to produce accurate and representative text samples. Given the potential to enhance our understanding and even aid in prevention efforts, the exploration of machine learning models, specifically in the context of synthetic data generation, holds promise. This novel approach offers a potentially powerful tool in our ongoing efforts to understand and counteract such societal issues, making it an area deserving of further investigation.

Lastly, data making up the portion of posts written by people belonging to the majority

class — regular people with no traits akin to those of a school shooter — could be extended by incorporating data from multiple sources. The Blogs Authorship corpus, previously used in several studies included in our literature review such as Neuman et al. (2015), Tschantret (2021), and Neuman et al. (2022), was initially collected for this research but was later excluded due to the risk of oversaturating the dataset with the majority class. However, this corpus, along with other widely used data sources such as the Twitter API, Reddit API, or the Yelp Open dataset, could be utilized to ensure the same ratio if the minority class were to increase drastically.

8.2.2 Utilizing Annotated Personality Scores

As mentioned in Subsection 7.1.3, our initial literature review led us to consider a personality prediction model as a potential solution. However, due to insufficient data and limitations in our capacity to generate robust data, this idea was eventually abandoned. Nevertheless, multiple studies exploring personality prediction methods grounded in models such as the Five-Factor Model or Myers-Briggs Type Indicator (MBTI) suggest potential promise in this approach (Markovikj et al., 2013; Lima & de Castro, 2014; Peng et al., 2015; Tandera et al., 2017), with an ability to quite precisely model personality for general purposes. This implies that, given an adequate volume of appropriately labeled data, a personality prediction model might indeed prove valuable when applied to this specific task of detecting school shooters as well.

Two potential directions for promising future research could be unveiled. The first involves the contribution of individuals with expertise in personality annotation, who could help annotate a substantial amount of data related to school shooters. If this data becomes publicly accessible, at least for research purposes, it could stimulate further investigation into utilizing these features for the automated classification of school shooters. The second area where future research is warranted lies in the development of a gold standard model that could automatically apply personality scores across multiple frameworks on a variety of text types. As a starting point, the proposed model could be trained using the existing personality score annotated and validated data, as there exists a considerable amount already, some of which have been referenced in this thesis. Given the broad array of classification models that could be applied to this task, it is plausible to hypothesize that a model capable of predicting personality scores with accuracy comparable to that of human experts could exist, provided it is trained on sufficiently high-quality data.

As for the selection of frameworks specifically, several options can be considered, including the aforementioned personality models, as well as specialized threat assessment protocols such as ERG22+ (Powis et al., 2019), VERA-2R (Pressman & Flockton, 2012; Pressman et al., 2016), MLG (Cook, 2014), and the TRAP-18 (Meloy & Gill, 2016), among others, as mentioned in Gill et al. (2020). These protocols have been widely utilized in forensic and law enforcement contexts to assess the risk posed by terrorists and violent extremists. If the rankings can be transformed into machine-readable scores, there is a possibility that these protocols could be applied to predict school shootings as well. An alternative approach could involve examining the Dark Triad (Paulhus & Williams, 2002), which encompasses the personality traits of narcissism, Machiavellianism, and psychopathy. By exploring the potential connection between these dark traits and the personalities of individuals who have perpetrated mass shootings, it may be possible to establish a correlation (Bushman, 2017; Bushman et al., 2018). Considering the array of instruments already utilized in related domains, employing the same tools for the challenge of automated school shooter identification could yield beneficial results in terms of predictive accuracy.

8.2.3 Further Work on Feature Sets and Combinations of These

In this thesis, the choice of text analysis framework for creating features trying to encapsulate personality was the tool of Linguistic Inquiry and Word Count. We opted to utilize this along with word embeddings, given its use in studies most closely aligned with our research direction. This comparison allowed for a more equitable assessment. However, there exist other frameworks similar to LIWC that could have been further explored. Two notable options found in the literature are Empath and SPLICE. Empath (Fast et al., 2016) is a text analysis tool that enables users to create and validate new categories on-demand, boasting a prevalidated set of 200 emotional and topical categories. It appears to rival LIWC and even compares with it directly in their paper, with additional contenders being ANew (Bradley & Lang, 1999), SentiWordNet (Esuli & Sebastiani, 2006), and EmoLex (Mohammad & Turney, 2010). The second candidate, Structured Programming for Linguistic Cue Extraction, used by Tandera et al. (2017) and Tadesse et al. (2018), appears quite similar to LIWC in functionality. Additionally, a more extensive application of Principal Component Analysis to LIWC categories could have been implemented. Although our emphasis was placed on a deeper exploration of word embeddings, based on the findings in related work and early experimentation, adopting the PCA methodology outlined by Kaati et al. (2016) could have yielded valuable results, given the impressive performance on their dataset of lone wolf perpetrators. However, it's important to note that their approach involved a smaller subset of school shooter texts, although more extensive texts, including mass shooter manifestos that range from hundreds to thousands of pages long. Regardless, this highlights the fact that there are numerous potential frameworks, feature sets, and combinations of these available for further exploration.

Bibliography

Arnoux, P.-H., Xu, A., Boyette, N., Mahmud, J., Akkiraju, R., & Sinha, V. (2017). 25 tweets to know you: A new model to predict personality with social media. https://doi.org/10.48550/arXiv.1704.05513

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *Proceedings of the International Conference on Learning Representations (ICLR)*. https://doi.org/10.48550/arXiv.1409.0473

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, *13*, 281–305. https://doi.org/10.5555/2188385.2188395

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching word vectors with subword information. https://doi.org/10.48550/arXiv.1607.04606

Bradley, M. M., & Lang, P. J. (1999). Affective Norms for English Words (ANEW): Instruction manual and affective ratings. Retrieved 3rd June 2023, from https://www.semanticscholar.org/paper/Affective-Norms-for-English-Words-(ANEW)%5C%3A-Manual-Bradley-Lang/c765eb0a31849361d829b24e173a37bab0919892

Breiman, L. (2001). Random Forests. *Machine learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Bushman, B. J. (2017). Narcissism, fame seeking, and mass shootings. *American Behavioral Scientist*, *62*. https://doi.org/10.1177/0002764217739660

Bushman, B. J., Coyne, S. M., Anderson, C. A., Björkqvist, K., Boxer, P., Dodge, K. A., Dubow, E. F., Farrington, D. P., Gentile, D. A., Huesmann, L. R., Lansford, J. E., Novaco, R. W., Ostrov, J. M., Underwood, M. K., Warburton, W. A., & Ybarra, M. L. (2018). Risk factors for youth violence: Youth violence commission, International Society for Research on Aggression (ISRA). *Aggressive Behavior*, *44*(4), 331–336. https://doi.org/10.1002/ab.21766

Cadwalladr, C., & Graham-Harrison, E. (2018). Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. *The Guardian*. Retrieved 2nd June 2023, from https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election

Celli, F., Pianesi, F., Stillwell, D., & Kosinski, M. (2013). Workshop on computational personality recognition: Shared task. *Proceedings of the International AAAI Conference on Web and Social Media*, *7*(2). https://doi.org/10.1609/icwsm.v7i2.14467

Chen, T., & Guestrin, C. (2016). XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2939672.2939785

Cook, A. N. (2014). *Risk assessment and management of group-based violence* (Doctoral dissertation). Simon Fraser University. Retrieved 28th May 2023, from https://core.ac.uk/download/pdf/56378418.pdf

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. https://doi.org/10.18653/v1/N19-1423

Ekwunife, N. E. (2022). *National security through social media intelligence: Domestic incident prediction* (Doctoral dissertation) [Access was given by Donna Schaeffer, Ph.D., committee chair]. Marymount University. Retrieved 10th November 2022, from https://www.proquest.com/openview/47a13c5fc4a34bc47135c2998cd7d94d/

Esuli, A., & Sebastiani, F. (2006). SENTIWORDNET: A publicly available lexical resource for opinion mining. *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. Retrieved 29th May 2030, from http://www.lrec-conf.org/proceedings/lrec2006/pdf/384_pdf.pdf

Faife, C. (2022). After Uvalde, social media monitoring apps struggle to justify surveillance. *The Verge*. Retrieved 1st June 2023, from https://www.theverge.com/2022/5/31/23148541/digital-surveillance-school-shootings-social-sentinel-uvalde

Farnadi, G., Zoghbi, S., Moens, M.-F., & De Cock, M. (2013). Recognising personality traits using Facebook status updates. https://doi.org/10.1609/icwsm.v7i2.14470

Fast, E., Chen, B., & Bernstein, M. S. (2016). Empath: Understanding topic signals in large-scale text. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/2858036.2858535

Gill, P., Marchment, Z., Zolghadriha, S., Salman, N., Rottweiler, B., Clemmow, C., & Vegt, I. v. d. (2020). Advances in violent extremist risk analysis. In D. M. D. Silva & M. Deflem (Eds.), *Radicalization and counter-radicalization* (pp. 55–74). Emerald Publishing Limited. https://doi.org/10.1108/S1521-613620200000025004

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. Retrieved 5th November 2022, from http://www.deeplearningbook.org

Hamm, M., & Spaaij, R. (2015). Lone wolf terrorism in America: Using knowledge of radicalization pathways to forge prevention strategies. *Washington, DC: US Department of Justice*. https://doi.org/doi.org/10.3886/ICPSR36107.v1

Harris, Z. S. (1954). Distributional structure. *WORD*, *10*(2-3), 146–162. https://doi.org/10.1080/00437956.1954.11659520

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., & Scholkopf, B. (1998). Support Vector Machines. *IEEE Intelligent Systems and their applications*, *13*(4), 18–28. https://doi.org/10.1109/5254.708428

Ho, T. K. (1995). Random decision forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition*, *1*, 278–282. https://doi.org/10.1109/ICDAR.1995.598994

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Internet Archive. (2020). The complete works of Randy Stair. Retrieved 12th February 2023, from https://archive.org/details/RandyStair

Junyi, S. (2020). Jieba Chinese text segmentation. Retrieved 5th December 2022, from https://github.com/fxsjy/jieba

Kaati, L., Shrestha, A., & Sardella, T. (2016). Identifying warning behaviors of violent lone offenders in written communication, 1053–1060. https://doi.org/10.1109/ICDMW.2016.0152

Kofod-Petersen, A. (2018). How to do a structured literature review in computer science. Retrieved 15th September 2022, from https://www.researchgate.net/publication/265158913_How_to_do_a_Structured_Literature_Review_in_computer_science

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). ALBERT: A lite BERT for self-supervised learning of language representations. https://doi.org/10.48550/arXiv.1909.11942

Langman, P. (2022). School shooters .info: Resources on school shootings, perpetrators, and prevention. Retrieved 22nd January 2023, from https://schoolshooters.info

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J. E., & Stoica, I. (2018). Tune: A research platform for distributed model selection and training. https://doi.org/10.48550/arXiv.1807.05118

Lima, A. C. E., & de Castro, L. N. (2014). A multi-label, semi-supervised classification approach applied to personality prediction in social media. *Neural Networks*, *58*, 122–130. https://doi.org/10.1016/j.neunet.2014.05.020

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized bert pretraining approach. https://doi.org/10.48550/arXiv.1907.11692

Loper, E., & Bird, S. (2002). NLTK: The Natural Language Toolkit. https://doi.org/10.48550/arXiv.cs/0205028

Mairesse, F., Walker, M., Mehl, M., & Moore, R. (2007). Using linguistic cues for the automatic recognition of personality in conversation and text. *J. Artif. Intell. Res. (JAIR)*, *30*, 457–500. https://doi.org/10.1613/jair.2349

Markovikj, D., Gievska, S., Kosinski, M., & Stillwell, D. (2013). Mining Facebook data for predictive personality modeling. https://doi.org/10.1609/icwsm.v7i2.14466

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133. https://doi.org/10.1007/BF02478259

McKinney, W. (2010). Data structures for statistical computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in science conference: SciPy 2010* (pp. 56–61). SciPy. https://doi.org/10.25080/Majora-92bf1922-00a

Meloy, J., & Gill, P. (2016). The lone-actor terrorist and the TRAP-18. *Journal of Threat Assessment and Management*, *3*, 37–52. https://doi.org/10.1037/tam0000061

Meta. (2023). Meta reports first quarter 2023 results. Retrieved 26th May 2023, from https://s21.q4cdn.com/399680738/files/doc_news/Meta-Reports-First-Quarter-2023-Results-2023.pdf

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. https://doi.org/10.48550/arXiv.1301.3781

Minsky, M., & Papert, S. (1969). *Perceptrons: An introduction to computational geometry*. MIT Press. https://doi.org/10.7551/mitpress/11301.001.0001

Mitchell, T. M. (1997). *Machine learning* (1st ed.). McGraw-Hill Professional.

Mohammad, S., & Turney, P. (2010). Emotions evoked by common words and phrases: Using Mechanical Turk to create an emotion lexicon. *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, 26–34. https://doi.org/10.5555/1860631.1860635

Mulyar, A., Schumacher, E., Rouhizadeh, M., & Dredze, M. (2019). Phenotyping of clinical notes with improved document classification models using contextualized neural language models. https://doi.org/10.48550/arXiv.1910.13664

National Threat Assessment Center. (2019). Protecting America's schools: A U.S. Secret Service analysis of targeted school violence. Retrieved 24th May 2023, from https://www.secretservice.gov/sites/default/files/2020-04/Protecting_Americas_Schools.pdf

Neuman, Y., Assaf, D., Cohen, Y., & Knoll, J. L. (2015). Profiling school shooters: Automatic text-based analysis. *Frontiers in Psychiatry*, 86. https://doi.org/10.3389/fpsyt.2015.00086

Neuman, Y., & Cohen, Y. (2014). A vectorial semantics approach to personality assessment. https://doi.org/10.1038/srep04761

Neuman, Y., Erez, E. S., Tschantret, J., & Weiss, H. (2022). Themes of revenge: Automatic identification of vengeful content in textual data. https://doi.org/10.48550/arXiv.2205.01731

Neuman, Y., Lev-Ran, Y., & Erez, E. S. (2020). Screening for potential school shooters through the Weight of Evidence. *Heliyon*, *6*. https://doi.org/10.1016/j.heliyon.2020.e05066

O'Leary, L. (2022). Why expensive social media monitoring has failed to protect schools. *Slate*. Retrieved 1st June 2023, from https://slate.com/technology/2022/06/social-media-monitoring-software-schools-safety.html

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. https://doi.org/10.48550/arXiv.1511.08458

Palomino, M., & Aider, F. (2022). Evaluating the effectiveness of text pre-processing in sentiment analysis. *Applied Sciences*, *12*, 8765. https://doi.org/10.3390/app12178765

Park, G., Schwartz, H. A., Eichstaedt, J. C., Kern, M. L., Kosinski, M., Stillwell, D. J., Ungar, L. H., & Seligman, M. E. P. (2014). Automatic personality assessment through social media language. *Journal of Personality and Social Psychology*. https://doi.org/10.1037/pspp0000020

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems 32* (pp. 8024–8035). arXiv. https://doi.org/10.48550/arXiv.1912.01703

Paulhus, D. L., & Williams, K. M. (2002). The Dark Triad of personality: Narcissism, Machiavellianism, and psychopathy. *Journal of Research in Personality*, *36*(6), 556–563. https://doi.org/10.1016/S0092-6566(02)00505-6

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2018). Scikit-learn: Machine learning in Python. https://doi.org/10.48550/arXiv.1201.0490

Peng, K.-H., Liou, L.-H., Chang, C.-S., & Lee, D.-S. (2015). Predicting personality traits of Chinese users based on Facebook wall posts. https://doi.org/10.1109/WOCC.2015.7346106

Pennebaker, J. W., & King, L. A. (1999). Linguistic styles: Language use as an individual difference. *Journal of Personality and Social Psychology*, *77*(6), 1296–1312. https://doi.org/10.1037/0022-3514.77.6.1296

Pennebaker, J. W., Francis, M. E., & Booth, R. J. (2001). Linguistic Inquiry and Word Count: LIWC 2001. Retrieved 12th March 2023, from https://www.researchgate.net/publication/239667728_Linguistic_Inquiry_and_Word_Count_LIWC_LIWC2001

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. https://doi.org/10.3115/v1/D14-1162

Plank, B., & Hovy, D. (2015). Personality traits on Twitter—or—How to get 1,500 personality tests in a week, 92–98. https://doi.org/10.18653/v1/W15-2913

Potts, C. (2011). Happyfuntokenizing.py. Retrieved 9th December 2022, from http://sentiment.christopherpotts.net/code-data/happyfuntokenizing.py

Powis, B., Randhawa, K., & Bishopp, D. (2019). An examination of the structural properties of the Extremism Risk Guidelines (ERG22+): A structured formulation tool for extremist offenders. *33*(6), 1141–1159. https://doi.org/10.1080/09546553.2019.1598392

Pratama, B. Y., & Sarno, R. (2015). Personality classification based on Twitter text using Naive Bayes, KNN and SVM, 170–174. https://doi.org/10.1109/ICODSE.2015.7436992

Pressman, D. E., & Flockton, J. (2012). Calibrating risk for violent political extremists and terrorists: The VERA 2 structured assessment. *The British Journal of Forensic Practice*, *14*(4), 237–251. https://doi.org/10.1108/14636641211283057

Pressman, D. E., Flockton, J., Rinne, T., & Duits, N. (2016). Violent Extremism Risk Assessment, version 2-revised (VERA-2R). Retrieved 27th May 2023, from https://home-affairs.ec.europa.eu/networks/radicalisation-awareness-network-ran/collection-inspiring-practices/ran-practices/violent-extremism-risk-assessment-version-2-revised-vera-2r-pressman-rinne-duits-flockton-2016_en

Ramezani, M., Feizi-Dekhshi, M.-R., Balafar, M.-A., Asgari-Chenaghlu, M., Feizi-Derakhshi, A.-R., Nikzad-Khasmakhi, N., Ranjbar-Khadivi, M., Jahanbakhsh-Nagadeh, Z., Zafarani-Moattar, E., & Rahkar-Farshi, R. (2018). Automatic personality prediction; an enhanced method using ensemble modeling. https://doi.org/10.48550/arXiv.2007.04571

Rasmussen, C. E., & Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. The MIT Press. https://doi.org/10.7551/mitpress/3206.001.0001

Riedman, D. (2023). K-12 school shooting database. Retrieved 20th May 2023, from https://k12ssdb.org/all-shootings

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386. https://doi.org/doi.org/10.1037/h0042519

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Sanders Analytics. (2013). Twitter corpus. Retrieved 8th December 2022, from https://github.com/zfz/twitter_corpus

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2020). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. https://doi.org/10.48550/arXiv.1910.01108

Semenov, A., Veijalainen, J., & Kyppo, J. (2010). Analysing the presence of school-shooting related communities at social media sites. *International Journal of Multimedia Intelligence and Security*, *1*(3). https://doi.org/10.1504/IJMIS.2010.037540

Shuster, M., & Nakajima, K. (2012). Japanese and Korean word search. https://doi.org/10.1109/ICASSP.2012.6289079

Simons, A., & Meloy, J. R. (2017). Foundations of threat assessment and management, 627–644. https://doi.org/10.1007/978-3-319-61625-4_36

Själander, M., Jahre, M., Tufte, G., & Reissmann, N. (2019). EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure. https://doi.org/10.48550/arXiv.1912.05848

Spärck Jones, K. (2004). A statistical interpretation of term specificity in retrieval. *Journal of Documentation*, *60*, 493–502. https://doi.org/10.1108/00220410410560573

Statista. (2023). Most popular social networks worldwide as of January 2023, ranked by number of monthly active users. Retrieved 30th May 2023, from https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

Stillwell, D., & Kosinski, M. (2015). myPersonality project website. Retrieved 25th May 2023, from https://sites.google.com/michalkosinski.com/mypersonality

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. https://doi.org/10.48550/arXiv.1409.3215

Tadesse, M. M., Lin, H., Xu, B., & Yang, L. (2018). Personality predictions based on user behavior on the Facebook social media platform. *IEEE Access*, *6*. https://doi.org/10.1109/ACCESS.2018.2876502

Tandera, T., Hendro, Suhartono, D., Wongso, R., & Prasetio, Y. L. (2017). Personality prediction system from Facebook users. *Procedia Computer Science*, *116*, 604–611. https://doi.org/10.1016/j.procs.2017.10.016

Tschantret, J. (2021). The psychology of right-wing terrorism: A text-based personality analysis. *Psychology of Violence*, *11*, 113–122. https://doi.org/10.1037/vio0000362

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6000–6010. https://doi.org/10.48550/arXiv.1706.03762

Vision of Humanity. (2019). Global Terrorism Index: The rise of the self-radicalised lone wolf terrorist. Retrieved 4th December 2022, from https://www.visionofhumanity.org/increase-in-self-radicalised-lone-wolf-attackers/

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering, 1–10. https://doi.org/10.1145/2601248.2601268

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., ... Rush, A. M. (2020). HuggingFace's Transformers: State-of-the-art natural language processing. https://doi.org/10.48550/arXiv.1910.03771

Wong, J. C. (2019). The Cambridge Analytica scandal changed the world – but it didn't change Facebook. *The Guardian*. Retrieved 2nd June 2023, from https://www.theguardian.com/technology/2019/mar/17/the-cambridge-analytica-scandal-changed-the-world-but-it-didnt-change-facebook

Zhang, H. (2004). The optimality of Naive Bayes. *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004*, *2*, 562–567. Retrieved 28th May 2023, from https://www.researchgate.net/publication/221439320_The_Optimality_of_Naive_Bayes

Appendices

A Primary and Secondary Inclusion Criteria

Q1:
Primary Inclusion Criteria

- The study's main concern is predicting/screening people capable of performing a school shooting.
- The study is a primary study presenting empirical results.

Secondary Inclusion Criteria

- The study focuses on finding people capable of performing a school shooting based on written social media activity

Q2:
Primary Inclusion Criteria

- The study's main concern is the prediction of personality.
- The study is a primary study presenting empirical results.

Secondary Inclusion Criteria

- The study focuses on predicting personality traits based on written social media activity.

B Quality Assessment Criteria

1. Is there a clear statement of the aim of the research?
2. Is the study put into the context of other studies and research?
3. Are system or algorithmic design decisions justified?
4. Is the test data set reproducible?
5. Is the study algorithm reproducible?
6. Is the experimental procedure thoroughly explained and reproducible?
7. Is it clearly stated in the study which other algorithms the study's algorithm(s) have been compared with?
8. Are the performance metrics used in the study explained and justified?
9. Are the test results thoroughly analyzed?
10. Does the test evidence support the findings presented?

C Primary Studies from the Literature Review

Id	Title	Author(s)
1	Identifying Warning Behaviors of Violent Lone Offenders in Written Communication	L. Kaati, A. Shrestha & T. Sardella
2	Profiling School Shooters: Automatic Text-Based Analysis	Y. Neuman, D. Assaf, Y. Cohen & J. L. Knoll
3	A Multi-Label, Semi-Supervised Classification Approach Applied to Personality Prediction in Social Media	A. C. E. S. Lima & L. N. Castro
4	Personality Predictions Based on User Behavior on the Facebook Social Media Platform	M. M. Tadesse, H. Lin, B. Xu & L. Yang
5	Personality Classification Based on Twitter Text Using Naive Bayes, KNN and SVM	B. Y. Pratama & R. Sarno
6	Recognising Personality Traits Using Facebook Status Updates	G. Farnadi, S. Zoghbi, M.-F. Moens & M. De Cock
7	Predicting Personality Traits of Chinese Users Based on Facebook Wall Posts	K.-H. Peng, L.-H. Liou, C.-S. Chang & D.-S. Lee
8	Personality Prediction System from Facebook Users	T. Tandera, H. Derwin, S. Rini, W. Yen & L. Prasetio
9	Mining Facebook Data for Predictive Personality Modeling	D. Markovikj, S. Gievska, M. Kosinski & D. Stillwell
10	Automatic Personality Assessment Through Social Media Language	G. Park, H. A. Schwartz, J. C. Eichstaedt, M. L. Kern, M. Kosinski, D. J. Stillwell, L. H. Ungar & M. E. P. Seligman
11	Personality Traits on Twitter or How to Get 1,500 Personality Tests in a Week	B. Plank & D. Hovy
12	25 Tweets to Know You: A New Model to Predict Personality With Social Media	P.-H. Arnoux, A. Xu, N. Boyette, J. Mahmud, R. Akkiraju & V. Sinha

Table C.1: List of studies returned from literature search

D Supplementary Papers from the Literature Review

Title	Author(s)	Year
How to (Better) Find a Perpetrator in a Haystack	Y. Neuman, Y. Cohen & Y. Neuman	2019
Empath: Understanding Topic Signals in Large-Scale Text	E. Fast, B. Chen & M. S. Bernstein	2016
Themes of Revenge: Automatic Identification of Vengeful Content in Textual Data	Y. Neuman, E. S. Erez, J. Tschantret & H. Weiss	2022
Visualizing the Relationship Among Indicators for Lone Actor Terrorist Attacks: Multidimensional Scaling and the TRAP-18	A. Goodwill & J. R. Meloy	2019
TRAP-18 Indicators Validated Through the Forensic Linguistic Analysis of Targeted Violence Manifestos.	J. Kupper & J. R. Meloy	2021
Detecting Linguistic Markers of Violent Extremism in Online Environments	F. Johansson, L. Kaati & M. Sahlgren	2017
Detecting Linguistic Markers for Radical Violence in Social Media	K. Cohen, F. Johansson, L. Kaati & J. C. Mork	2014
Linguistic Analysis of Lone Offender Manifestos	L. Kaati, A. Shrestha & K. Cohen	2016
Assessing Violence Risk in Threatening Communications	K. Glasgow & R. Schouten	2014
The Role of Warning Behaviors in Threat Assessment: An Exploration and Suggested Typology	J. R. Meloy, J. Hoffmann, A. Guldimann & D. James	2012
A Linguistic Analysis of Mass Shooter Journals, Diaries, Correspondence, and Manifestos	H. Duong	2020
Advances in Violent Extremist Risk Analysis	P. Gill, Z. Marchment, S. Zolghadriha, N. Salman, B. Rottweiler, C. Clemmow & I. V. D. Vegt	2020
Analysis of Weak Signals for Detecting Lone Wolf Terrorists	J. Brynielsson, A. Horndahl, F. Johansson, L. Kaati, C. Mårtenson & P. Svenson	2012
Assessment of Risk in Written Communication Introducing the Profile Risk Assessment Tool (PRAT)	N. Akrami, A. Shrestha, M. Berggren, L. Kaati & M. Obaidi	2018

Table D.1: List of 14 supplementary papers after applying the snowballing method

E LIWC Categories

Category	Description
Affiliation	Words related to social and familial connections.
Allpunc	The count of all punctuation marks in the text.
Allure	Words related to attraction, appeal, and charm.
Analytic	A measure of the formal, logical, and hierarchical thinking in a text.
Authentic	A measure of the authenticity or honesty of the text.
Big Words	Words with more than six letters.
Clout	A measure of the authoritativeness or confidence of the text.
Cognition	Related to cognitive processes, such as thinking and problem-solving.
Communication	Words related to interaction, dialogue, and communication.
Compare	Words used to make comparisons (e.g., greater, less, as).
Conflict	Words related to disagreement, conflict, and struggle.
Curiosity	Words related to interest, curiosity, and inquisitiveness.
Emo_pos	Words related to positive emotions, such as love, nice, sweet.
Exclam	The count of exclamation marks (!) in the text.
Fatigue	Words related to tiredness, exhaustion, and fatigue.
Friends	Words related to friends and companionship.
Function	Function words, which include pronouns, prepositions, auxiliary verbs, conjunctions, and articles.
Humans	Words specifically referring to human beings.
Insight	Words related to understanding or revelation (e.g., think, know).
Lifestyle	Words related to social, personal, leisure, home, and money matters.
Memory	Words related to remembrance, nostalgia, and memory.
Mental	Words related to cognitive processes and mental states.
Moral	Words related to morality, ethics, and values.
Need	Words expressing requirement, necessity, or desire.
Otherp	The count of other punctuations (excluding period, comma, colon, semicolon, question mark, and exclamation mark) in the text.
Period	The count of periods (.) in the text.
Politics	Words related to political processes, issues, and ideologies.
Ppron	Personal pronouns, including first, second, and third person pronouns.
Pronoun	Words that take the place of a noun or noun phrase (e.g., he, she, it).
Reward	Words related to rewards, success, and achievements.
Risk	Words related to danger, risk, and uncertainty.
Social	Words referring to other people and social processes.
Socrefs	Social references, usually measured as the use of social words.
Swear Words	Profane, vulgar, or offensive language.
Time	Words related to time, including past, present, and future tenses.
Tone_pos	A measure of the positive emotional tone of the text.
Tone	A measure of the emotional tone of the text, with higher values indicating a more positive tone.
WC (Word Count)	The total number of words in a text.
Want	Words expressing desire, longing, or wish.
Wellness	Words related to health and well-being.
Work	Words related to work, job, and career.
You	Second person pronouns (you, your, yours, etc.).

Table E.1: Glossary of mentioned LIWC categories

F Hyperparameter Configurations

F.1 Scikit-Learn Models on LIWC

Naïve Bayes (NB)

LIWC Version	Max Length	Parameters
2001	256	-
2001	512	-
2007	256	-
2007	512	-
2015	256	-
2015	512	-
2022	256	-
2022	512	-

Table F.1: Final hyperparameters per LIWC version and max length for NB

K-Nearest Neighbors (KNN)

LIWC Version	Max Length	Metric	N_Neighbors
2001	256	euclidean	1
2001	512	euclidean	1
2007	256	manhattan	1
2007	512	manhattan	1
2015	256	manhattan	1
2015	512	manhattan	1
2022	256	manhattan	1
2022	512	manhattan	1

Table F.2: Final hyperparameters per LIWC version and max length for KNN

Support Vector Machine (SVM)

LIWC Version	Max Length	C	Gamma	Kernel
2001	256	100	scale	rbf
2001	512	100	scale	rbf
2007	256	100	scale	rbf
2007	512	100	scale	rbf
2015	256	100	scale	rbf
2015	512	100	scale	rbf
2022	256	100	scale	rbf
2022	512	100	scale	rbf

Table F.3: Final hyperparameters per LIWC version and max length for SVM

Gaussian Processes (GP)

LIWC Version	Max Length	Kernel
2001	256	RBF(length_scale=1) + WhiteKernel(noise_level=1)
2001	512	RBF(length_scale=1) + WhiteKernel(noise_level=1)
2007	256	RBF(length_scale=1) + WhiteKernel(noise_level=1)
2007	512	RBF(length_scale=1) + WhiteKernel(noise_level=1)
2015	256	RBF(length_scale=1) + WhiteKernel(noise_level=1)
2015	512	RBF(length_scale=1) + WhiteKernel(noise_level=1)
2022	256	RBF(length_scale=1) + WhiteKernel(noise_level=1)
2022	512	RBF(length_scale=1) + WhiteKernel(noise_level=1)

Table F.4: Final hyperparameters per LIWC version and max length for GP

Extreme Gradient Boosting (XGBoost)

LIWC Version	Max Length	Learning Rate	N_Estimators
2001	256	0.1	300
2001	512	0.1	300
2007	256	0.1	300
2007	512	0.1	300
2015	256	0.1	300
2015	512	0.1	300
2022	256	0.1	300
2022	512	0.1	300

Table F.5: Final hyperparameters per LIWC version and max length for XGBoost

F.2 Scikit-Learn Models on Word Embeddings

Naïve Bayes (NB)

Emb. Type	Max Length	Padding	Parameters
bert	256	head	-
bert	256	split	-
bert	256	tail	-
bert	512	head	-
bert	512	split	-
bert	512	tail	-
fasttext	256	head	-
fasttext	256	split	-
fasttext	256	tail	-
fasttext	512	head	-
fasttext	512	split	-
fasttext	512	tail	-
glove	256	head	-
glove	256	split	-
glove	256	tail	-
glove	512	head	-
glove	512	split	-
glove	512	tail	-
glove_50	256	head	-
glove_50	256	split	-
glove_50	256	tail	-
glove_50	512	head	-
glove_50	512	split	-
glove_50	512	tail	-

Table F.6: Final hyperparameters per combination of max length, embedding type, and padding position for NB on word embeddings

K-Nearest Neighbors (KNN)

Emb. Type	Max Length	Padding	Metric	N_Neighbors
bert	256	head	manhattan	1
bert	256	split	manhattan	1
bert	256	tail	manhattan	1
bert	512	head	manhattan	1
bert	512	split	manhattan	1
bert	512	tail	manhattan	1
fasttext	256	head	manhattan	1
fasttext	256	split	manhattan	1
fasttext	256	tail	manhattan	1
fasttext	512	head	manhattan	1
fasttext	512	split	manhattan	1
fasttext	512	tail	manhattan	1
glove	256	head	euclidean	1
glove	256	split	euclidean	1
glove	256	tail	euclidean	1
glove	512	head	euclidean	1
glove	512	split	euclidean	1
glove	512	tail	euclidean	1
glove_50	256	head	manhattan	1
glove_50	256	split	euclidean	1
glove_50	256	tail	euclidean	1
glove_50	512	head	manhattan	1
glove_50	512	split	euclidean	1
glove_50	512	tail	euclidean	1

Table F.7: Final hyperparameters per combination of max length, embedding type, and padding position for KNN on word embeddings

Support Vector Machine (SVM)

Emb. Type	Max Length	Padding	Kernel	C	Gamma
bert	256	head	sigmoid	100	scale
bert	256	split	sigmoid	100	scale
bert	256	tail	sigmoid	100	scale
bert	512	head	linear	0.1	scale
bert	512	split	sigmoid	100	scale
bert	512	tail	sigmoid	100	scale
fasttext	256	head	sigmoid	100	scale
fasttext	256	split	sigmoid	100	scale
fasttext	256	tail	sigmoid	100	scale
fasttext	512	head	sigmoid	100	scale
fasttext	512	split	sigmoid	100	scale
fasttext	512	tail	sigmoid	100	scale
glove	256	head	sigmoid	100	scale
glove	256	split	sigmoid	100	scale
glove	256	tail	linear	10	scale
glove	512	head	sigmoid	100	scale
glove	512	split	sigmoid	100	scale
glove	512	tail	linear	10	scale
glove_50	256	head	sigmoid	100	scale
glove_50	256	split	linear	100	scale
glove_50	256	tail	sigmoid	100	scale
glove_50	512	head	sigmoid	100	scale
glove_50	512	split	linear	0.1	scale
glove_50	512	tail	sigmoid	100	scale

Table F.8: Final hyperparameters per combination of max length, embedding type, and padding position for SVM on word embeddings

Gaussian Processes (GP)

Max Length	Emb. Type	Pad. Pos.	Kernel
bert	256	head	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
bert	256	split	RBF(length_scale=1) + WhiteKernel(noise_level=1)
bert	256	tail	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
bert	512	head	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
bert	512	split	RBF(length_scale=1) + WhiteKernel(noise_level=1)
bert	512	tail	RBF(length_scale=1) + WhiteKernel(noise_level=1)
fasttext	256	head	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
fasttext	256	split	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
fasttext	256	tail	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
fasttext	512	head	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
fasttext	512	split	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
fasttext	512	tail	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
glove	256	head	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
glove	256	split	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
glove	256	tail	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
glove	512	head	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
glove	512	split	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
glove	512	tail	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
glove_50	256	head	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
glove_50	256	split	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
glove_50	256	tail	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
glove_50	512	head	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
glove_50	512	split	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)
glove_50	512	tail	DotProduct(sigma_0=1) + WhiteKernel(noise_level=1)

Table F.9: Final hyperparameters per combination of max length, embedding type, and padding position for GP on word embeddings

Extreme Gradient Boosting (XGBoost)

Max Length	Embedding Type	Padding Position	Learning Rate	Number of Estimators	Gamma
bert	256	head	0.1	250	0
bert	256	split	0.1	250	0
bert	256	tail	0.1	250	0
bert	512	head	0.1	200	0
bert	512	split	0.01	50	0
bert	512	tail	0.1	250	0
fasttext	256	head	0.1	300	0
fasttext	256	split	0.1	300	0
fasttext	256	tail	0.1	300	0
fasttext	512	head	0.1	250	0
fasttext	512	split	0.1	150	0
fasttext	512	tail	0.1	250	0
glove	256	head	0.1	300	0
glove	256	split	0.1	150	0
glove	256	tail	0.1	250	0
glove	512	head	0.1	250	0
glove	512	split	0.1	250	0
glove	512	tail	0.1	300	0
glove_50	256	head	0.1	300	0
glove_50	256	split	0.1	300	0
glove_50	256	tail	0.1	300	0
glove_50	512	head	0.1	300	0
glove_50	512	split	0.1	250	0
glove_50	512	tail	0.1	200	0

Table F.10: Final hyperparameters per combination of max length, embedding type, and padding position for XGBoost on word embeddings

F.3 PyTorch Neural Networks on Word Embeddings

Bidirectional Long Short-Term Memory (biLSTM)

Embedding Type	Max Length	Pad. Pos.	Dropout	LR	Batch Size	Hidden Size	Layers
bert	256	head	0.6	0.0063	256	64	2
bert	256	split	0.6	0.0054	128	256	2
bert	256	tail	0.5	0.0002	256	256	3
bert	512	head	0.3	0.0011	128	256	1
bert	512	split	0.5	0.0052	256	64	1
bert	512	tail	0.4	0.0009	128	128	2
fasttext	256	head	0.3	0.0003	128	128	3
fasttext	256	split	0.4	0.0005	64	64	1
fasttext	256	tail	0.6	0.0034	128	64	1
fasttext	512	head	0.3	0.0006	64	256	1
fasttext	512	split	0.5	0.0079	64	64	3
fasttext	512	tail	0.4	0.0002	64	256	2
glove	256	head	0.4	0.0026	128	128	2
glove	256	split	0.4	0.0085	256	64	3
glove	256	tail	0.3	0.0334	256	64	1
glove	512	head	0.4	0.0002	128	128	3
glove	512	split	0.3	0.0144	256	64	1
glove	512	tail	0.6	0.0007	64	256	2
glove_50	256	head	0.4	0.0008	256	128	3
glove_50	256	split	0.6	0.0025	256	256	3
glove_50	256	tail	0.5	0.0008	256	64	3
glove_50	512	head	0.5	0.0075	256	64	3
glove_50	512	split	0.6	0.0035	256	256	1
glove_50	512	tail	0.4	0.0004	64	256	2

Table F.11: Final hyperparameters per combination of max length, embedding type, and padding position for biLSTM on word embeddings

Convolutional Neural Network (CNN)

Embedding Type	Max Length	Padding Position	Dropout	LR	Batch Size
bert	256	head	0.6	0.0002	64
bert	256	split	0.3	0.0001	64
bert	256	tail	0.3	0.0003	128
bert	512	head	0.3	0.0011	256
bert	512	split	0.4	0.0002	128
bert	512	tail	0.5	0.0010	256
fasttext	256	head	0.3	0.0049	128
fasttext	256	split	0.3	0.0005	64
fasttext	256	tail	0.4	0.0064	128
fasttext	512	head	0.3	0.0018	256
fasttext	512	split	0.3	0.0010	256
fasttext	512	tail	0.3	0.0003	128
glove	256	head	0.3	0.0005	128
glove	256	split	0.5	0.0010	128
glove	256	tail	0.3	0.0008	64
glove	512	head	0.3	0.0011	256
glove	512	split	0.3	0.0006	128
glove	512	tail	0.3	0.0009	256
glove_50	256	head	0.4	0.0042	256
glove_50	256	split	0.3	0.0009	128
glove_50	256	tail	0.4	0.0019	128
glove_50	512	head	0.3	0.0009	256
glove_50	512	split	0.5	0.0017	128
glove_50	512	tail	0.3	0.0011	128

Table F.12: Final hyperparameters per combination of max length, embedding type, and padding position for CNN on word embeddings

F.4 Hugging Face Transformer Models

Model	Max Length	Epochs	Batch Size	Learning Rate	Weight Decay
albert-base-v2	256	7	32	3e-05	0.0
albert-base-v2	512	5	32	2e-05	0.1
bert-base-uncased	256	7	32	3e-05	0.1
bert-base-uncased	512	7	32	2e-05	0.1
distilbert-base-uncased	256	7	32	3e-05	0.1
distilbert-base-uncased	512	7	32	3e-05	0.0
roberta-base	256	7	64	2e-05	0.0
roberta-base	512	10	32	2e-05	0.0

Table F.13: Final hyperparameters for Large Language Models

G Test Results

G.1 Scikit-Learn Models on LIWC

Model	Max Length 256				Max Length 512			
	Precision	**Recall**	**F_1**	**F_2**	**Precision**	**Recall**	**F_1**	**F_2**
nb	0.1828	**0.7788**	0.2961	0.4714	0.1865	**0.7965**	0.3022	**0.4815**
knn	0.4141	0.4515	0.4320	0.4435	0.4050	0.4447	0.4240	0.4362
svm	0.7020	0.4786	**0.5691**	**0.5111**	0.7095	0.4297	**0.5352**	0.4664
gaussian	0.7583	0.2054	0.3233	0.2405	**0.7596**	0.1985	0.3147	0.2329
xgboost	**0.8084**	0.3905	0.5266	0.4356	0.7209	0.3894	0.5057	0.4289

Table G.1: Performance scores for classical models on LIWC 2022

Model	Max Length 256				Max Length 512			
	Precision	**Recall**	**F_1**	**F_2**	**Precision**	**Recall**	**F_1**	**F_2**
nb	0.3041	**0.5327**	0.3872	0.4631	0.2419	**0.6357**	0.3504	**0.4795**
knn	0.4012	0.4537	0.4259	0.4422	0.4106	0.4673	0.4371	0.4548
svm	0.7054	0.4379	0.5404	**0.4739**	0.7286	0.3643	0.4858	0.4048
gaussian	0.8033	0.2212	0.3469	0.2587	**0.8039**	0.2060	0.3280	0.2420
xgboost	**0.8095**	0.4221	**0.5549**	0.4668	0.7551	0.3719	**0.4983**	0.4139

Table G.2: Performance scores for classical models on LIWC 2015

Model	Max Length 256				Max Length 512			
	Precision	**Recall**	**F_1**	**F_2**	**Precision**	**Recall**	**F_1**	**F_2**
nb	0.2763	**0.5576**	0.3695	0.4632	0.2355	**0.5729**	0.3338	**0.4453**
knn	0.4130	0.5147	0.4583	**0.4905**	0.3750	0.4523	0.4100	0.4344
svm	0.7312	0.4176	**0.5316**	0.4568	0.7500	0.3317	0.4599	0.3733
gaussian	**0.8013**	0.2822	0.4174	0.3242	**0.7881**	0.2337	0.3605	0.2719
xgboost	0.7544	0.3883	0.5127	0.4300	0.7512	0.3794	**0.5042**	0.4211

Table G.3: Performance scores for classical models on LIWC 2007

Model	Max Length 256				Max Length 512			
	Precision	**Recall**	**F_1**	**F_2**	**Precision**	**Recall**	**F_1**	**F_2**
nb	0.1913	**0.7562**	0.3054	0.4754	0.1896	**0.7814**	0.3052	**0.4811**
knn	0.3742	0.5169	0.4341	**0.4803**	0.3513	0.4749	0.4039	0.4437
svm	0.7600	0.3860	**0.5119**	0.4281	0.7762	0.2789	0.4103	0.3199
gaussian	**0.8201**	0.2573	0.3918	0.2983	**0.8137**	0.2085	0.3320	0.2450
xgboost	0.7742	0.3792	0.5091	0.4223	0.7097	0.3317	**0.4521**	0.3712

Table G.4: Performance scores for classical models on LIWC 2001

G.2 Scikit-Learn Models on Word Embeddings

BERT

Model	Max Length	Padding	Precision	Recall	F_1	F_2
nb	256	head	0.1632	0.7540	0.2683	0.4373
nb	256	split	0.1633	0.7540	0.2684	0.4374
nb	256	tail	0.1633	0.7540	0.2684	0.4374
nb	512	head	0.1619	**0.8417**	0.2716	0.4575
nb	512	split	0.1621	**0.8417**	0.2719	0.4579
nb	512	tail	0.1622	**0.8417**	0.2720	0.4580
knn	256	head	0.5000	0.2641	0.3456	0.2916
knn	256	split	0.5112	0.2054	0.2931	0.2333
knn	256	tail	0.6111	0.3228	0.4225	0.3564
knn	512	head	0.5200	0.2940	0.3756	0.3220
knn	512	split	0.5543	0.2563	0.3505	0.2872
knn	512	tail	0.6266	0.3668	0.4628	0.4000
svm	256	head	0.6014	0.6027	0.6020	0.6024
svm	256	split	0.5925	0.6433	0.6169	0.6325
svm	256	tail	0.6277	0.6546	0.6409	0.6491
svm	512	head	0.7169	0.6809	0.6985	0.6878
svm	512	split	0.7064	0.6709	0.6881	0.6777
svm	512	tail	0.7855	0.6809	0.7295	0.6995
gaussian	256	head	0.7848	0.7246	0.7535	**0.7359**
gaussian	256	split	0.6000	0.0203	0.0393	0.0252
gaussian	256	tail	0.8369	0.7066	**0.7662**	0.7293
gaussian	512	head	0.7839	0.6834	0.7302	0.7014
gaussian	512	split	0.5000	0.0176	0.0340	0.0218
gaussian	512	tail	0.5000	0.0176	0.0340	0.0218
xgboost	256	head	0.9199	0.5440	0.6837	0.5924
xgboost	256	split	0.9211	0.3950	0.5529	0.4460
xgboost	256	tail	0.9244	0.6072	0.7330	0.6520
xgboost	512	head	0.8828	0.5678	0.6911	0.6115
xgboost	512	split	**1.0000**	0.0980	0.1785	0.1196
xgboost	512	tail	0.9433	0.5854	0.7225	0.6335

Table G.5: Performance scores for classical models on BERT-embeddings

FastText

Model	Max Length	Padding	Precision	Recall	$\mathbf{F}_1$	$\mathbf{F}_2$
nb	256	head	0.2169	0.0406	0.0684	0.0485
nb	256	split	0.2195	0.0406	0.0686	0.0485
nb	256	tail	0.2048	0.0384	0.0646	0.0458
nb	512	head	0.2609	0.0151	0.0285	0.0186
nb	512	split	0.3044	0.0176	0.0333	0.0217
nb	512	tail	0.2917	0.0176	0.0332	0.0217
knn	256	head	0.2625	0.4018	0.3176	0.3633
knn	256	split	0.2724	0.3296	0.2983	0.3163
knn	256	tail	0.2862	0.3657	0.3211	0.3465
knn	512	head	0.2579	0.4724	0.3336	0.4050
knn	512	split	0.2400	0.3467	0.2837	0.3184
knn	512	tail	0.2783	0.3643	0.3156	0.3431
svm	256	head	0.4862	0.5169	0.5011	0.5105
svm	256	split	0.4989	0.5237	0.5110	0.5186
svm	256	tail	0.4408	0.4537	0.4472	0.4511
svm	512	head	0.4988	0.5377	0.5175	0.5294
svm	512	split	0.5128	**0.5528**	0.5320	0.5443
svm	512	tail	0.4988	0.5126	0.5056	0.5098
gaussian	256	head	0.6677	0.5124	0.5798	0.5374
gaussian	256	split	0.6422	0.4740	0.5455	0.5002
gaussian	256	tail	0.6626	0.4876	0.5618	0.5148
gaussian	512	head	0.6817	0.5327	**0.5980**	**0.5570**
gaussian	512	split	0.6268	0.4472	0.5220	0.4744
gaussian	512	tail	0.6871	0.4799	0.5651	0.5107
xgboost	256	head	0.8462	0.1987	0.3218	0.2345
xgboost	256	split	0.8072	0.1512	0.2548	0.1806
xgboost	256	tail	0.7661	0.2144	0.3351	0.2505
xgboost	512	head	0.8640	0.2714	0.4130	0.3145
xgboost	512	split	0.8182	0.1583	0.2653	0.1887
xgboost	512	tail	**0.8947**	0.2563	0.3984	0.2989

Table G.6: Performance scores for classical models on FastText-embeddings

GloVe with Embedding Size 300

Model	Max Length	Padding	Precision	Recall	F_1	F_2
nb	256	head	0.2198	0.0454	0.0752	0.0539
nb	256	split	0.2273	0.0454	0.0756	0.0540
nb	256	tail	0.2235	0.0431	0.0722	0.0514
nb	512	head	0.2200	0.0276	0.0491	0.0335
nb	512	split	0.2128	0.0251	0.0449	0.0305
nb	512	tail	0.2381	0.0251	0.0455	0.0306
knn	256	head	0.2314	0.3946	0.2917	0.3458
knn	256	split	0.2215	0.2948	0.2529	0.2765
knn	256	tail	0.2742	0.3787	0.3181	0.3519
knn	512	head	0.2407	0.4070	0.3025	0.3576
knn	512	split	0.2586	0.3593	0.3007	0.3333
knn	512	tail	0.3198	0.4146	0.3611	0.3914
svm	256	head	0.5180	0.4898	0.5035	0.4952
svm	256	split	0.5023	0.4966	0.4994	0.4977
svm	256	tail	0.4208	0.5057	0.4593	0.4861
svm	512	head	0.4801	0.4548	0.4671	0.4596
svm	512	split	0.5013	0.4849	0.4930	0.4881
svm	512	tail	0.4335	**0.5075**	0.4676	0.4908
gaussian	256	head	0.6742	0.4739	0.5566	0.5039
gaussian	256	split	0.6813	0.4943	0.5729	0.5230
gaussian	256	tail	0.6863	0.5011	**0.5793**	**0.5297**
gaussian	512	head	0.6537	0.4648	0.5433	0.4933
gaussian	512	split	0.6312	0.4774	0.5436	0.5018
gaussian	512	tail	0.6912	0.4950	0.5769	0.5248
xgboost	256	head	0.8624	0.2132	0.3418	0.2509
xgboost	256	split	0.8256	0.1610	0.2694	0.1919
xgboost	256	tail	0.8362	0.2200	0.3483	0.2580
xgboost	512	head	**0.8818**	0.2437	0.3819	0.2850
xgboost	512	split	0.8252	0.2136	0.3393	0.2507
xgboost	512	tail	0.8333	0.2638	0.4008	0.3056

Table G.7: Performance scores for classical models on GloVe-embeddings

GloVe with Embedding Size 50

Model	Max Length	Padding	Precision	Recall	$\mathbf{F}_1$	$\mathbf{F}_2$
nb	256	head	0.2500	0.0454	0.0768	0.0542
nb	256	split	0.2468	0.0431	0.0734	0.0516
nb	256	tail	0.2405	0.0431	0.0731	0.0516
nb	512	head	0.2444	0.0276	0.0497	0.0336
nb	512	split	0.2564	0.0251	0.0458	0.0307
nb	512	tail	0.2703	0.0251	0.0460	0.0307
knn	256	head	0.2841	0.2925	0.2883	0.2908
knn	256	split	0.2987	0.2540	0.2745	0.2618
knn	256	tail	0.3205	0.3197	0.3201	0.3199
knn	512	head	0.3061	0.3668	0.3337	0.3528
knn	512	split	0.3098	0.2864	0.2977	0.2908
knn	512	tail	0.3179	0.3442	0.3305	0.3386
svm	256	head	0.4186	0.3674	0.3913	0.3766
svm	256	split	0.4848	0.3243	0.3886	0.3473
svm	256	tail	0.4359	**0.3855**	0.4092	0.3946
svm	512	head	0.4174	0.3618	0.3876	0.3717
svm	512	split	0.6158	0.2739	0.3791	0.3081
svm	512	tail	0.4845	0.3141	0.3811	0.3378
gaussian	256	head	0.6388	0.3288	0.4341	0.3641
gaussian	256	split	0.6991	0.3583	0.4738	0.3970
gaussian	256	tail	0.6464	**0.3855**	**0.4830**	**0.4193**
gaussian	512	head	0.6650	0.3292	0.4403	0.3661
gaussian	512	split	0.6359	0.3116	0.4182	0.3469
gaussian	512	tail	0.6435	0.3493	0.4528	0.3844
xgboost	256	head	0.8583	0.2336	0.3672	0.2734
xgboost	256	split	0.8911	0.2041	0.3321	0.2413
xgboost	256	tail	0.7920	0.2245	0.3498	0.2620
xgboost	512	head	**0.9000**	0.2487	0.3898	0.2908
xgboost	512	split	0.7826	0.1809	0.2939	0.2138
xgboost	512	tail	0.8558	0.2236	0.3546	0.2624

Table G.8: Performance scores for classical models on GloVe-embeddings (dim 50)

G.3 PyTorch Neural Networks on Word Embeddings

BERT

Model	Max Length	Padding	Precision	Recall	F_1	F_2
biLSTM	256	head	0.7913	0.9074	0.8454	**0.8816**
biLSTM	256	split	0.7016	0.9074	0.7913	0.8571
biLSTM	256	tail	0.7118	0.9142	0.8004	0.8650
biLSTM	512	head	0.8668	0.8015	0.8329	0.8138
biLSTM	512	split	0.6986	0.8794	0.7786	0.8361
biLSTM	512	tail	0.7953	0.8492	0.8214	0.8379
CNN	256	head	0.8359	**0.8623**	0.8489	0.8569
CNN	256	split	0.8617	0.8578	**0.8597**	0.8586
CNN	256	tail	0.8571	0.8533	0.8552	0.8540
CNN	512	head	**0.8994**	0.8090	0.8519	0.8256
CNN	512	split	0.8608	0.8543	0.8575	0.8556
CNN	512	tail	0.8479	0.8543	0.8511	0.8530

Table G.9: Performance scores for neural networks on BERT-embeddings

FastText

Model	Max Length	Padding	Precision	Recall	F_1	F_2
biLSTM	256	head	0.5554	0.8281	0.6649	0.7540
biLSTM	256	split	0.4263	**0.9095**	0.5805	0.7414
biLSTM	256	tail	0.6504	0.7828	0.7105	0.7522
biLSTM	512	head	0.6023	0.7839	0.6812	0.7393
biLSTM	512	split	0.7036	0.7814	**0.7405**	**0.7645**
biLSTM	512	tail	0.6031	0.7940	0.6855	0.7467
CNN	256	head	0.5461	0.7511	0.6324	0.6987
CNN	256	split	0.6674	0.6991	0.6829	0.6925
CNN	256	tail	0.5951	0.7149	0.6495	0.6873
CNN	512	head	0.7150	0.7186	0.7168	0.7179
CNN	512	split	**0.7261**	0.7261	0.7261	0.7261
CNN	512	tail	0.6916	0.7211	0.7060	0.7150

Table G.10: Performance scores for neural networks on FastText-embeddings

GloVe with Embedding Dimension 300

Model	Max Length	Padding	Precision	Recall	$\mathbf{F}_1$	$\mathbf{F}_2$
biLSTM	256	head	0.4395	**0.9457**	0.6001	0.7687
biLSTM	256	split	0.6064	0.8575	0.7104	**0.7919**
biLSTM	256	tail	0.5373	0.8145	0.6475	0.7383
biLSTM	512	head	0.5413	0.8392	0.6581	0.7560
biLSTM	512	split	**0.6494**	0.7864	**0.7114**	0.7546
biLSTM	512	tail	0.5839	0.8216	0.6827	0.7598
CNN	256	head	0.6098	0.7285	0.6639	0.7012
CNN	256	split	0.5378	0.8054	0.6449	0.7325
CNN	256	tail	0.6288	0.7398	0.6798	0.7146
CNN	512	head	0.6175	0.7261	0.6674	0.7015
CNN	512	split	0.6398	0.7186	0.6769	0.7013
CNN	512	tail	0.6203	0.7387	0.6743	0.7115

Table G.11: Performance scores for neural networks on GloVe-embeddings

GloVe with Embedding Dimension 50

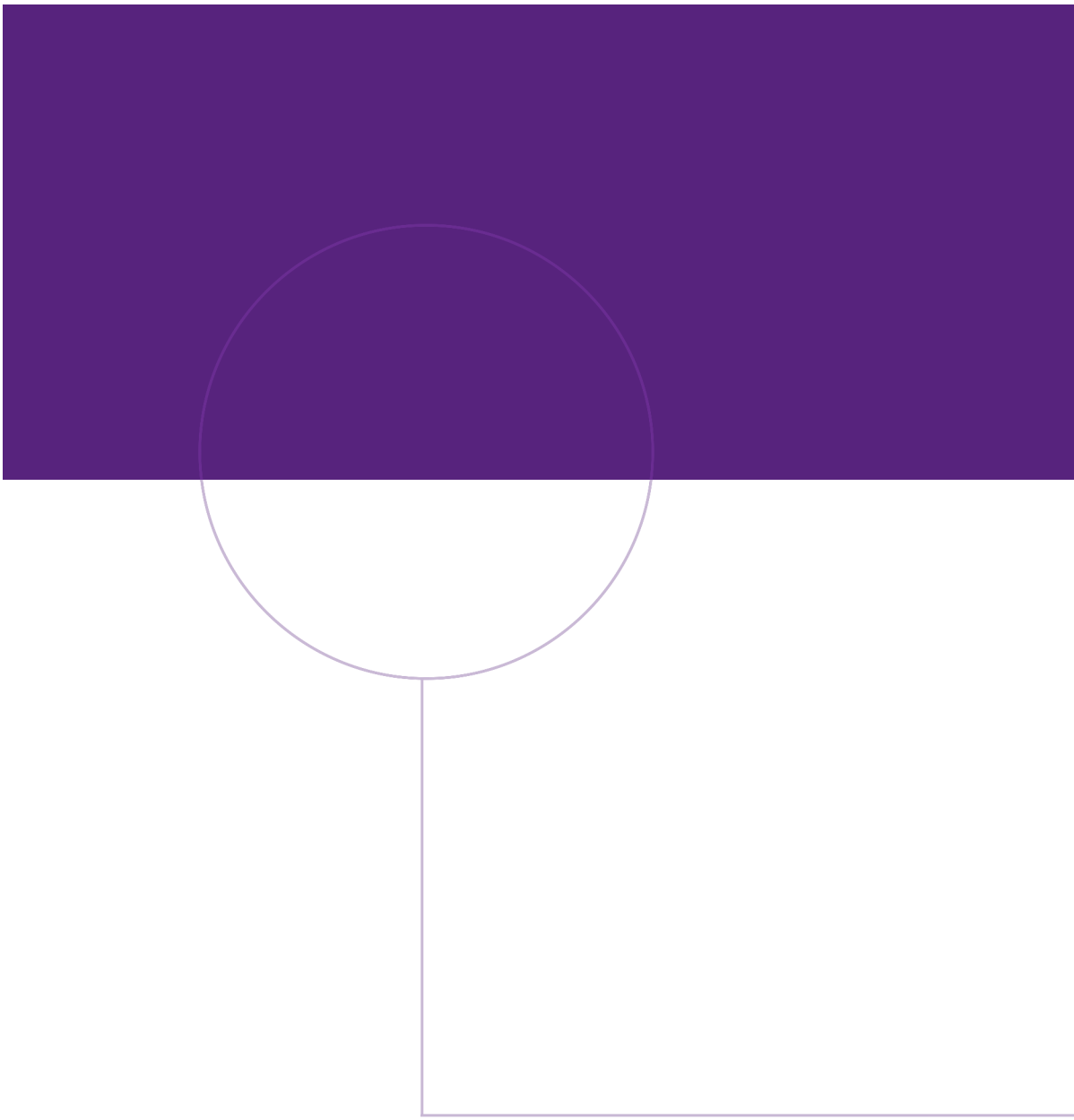
Model	Max Length	Padding	Precision	Recall	$\mathbf{F}_1$	$\mathbf{F}_2$
biLSTM	256	head	0.4395	**0.9457**	0.6001	0.7687
biLSTM	256	split	0.6064	0.8575	0.7104	**0.7919**
biLSTM	256	tail	0.5373	0.8145	0.6475	0.7383
biLSTM	512	head	0.5413	0.8392	0.6581	0.7560
biLSTM	512	split	**0.6494**	0.7864	**0.7114**	0.7546
biLSTM	512	tail	0.5839	0.8216	0.6827	0.7598
CNN	256	head	0.6098	0.7285	0.6639	0.7012
CNN	256	split	0.5378	0.8054	0.6449	0.7325
CNN	256	tail	0.6288	0.7398	0.6798	0.7146
CNN	512	head	0.6175	0.7261	0.6674	0.7015
CNN	512	split	0.6398	0.7186	0.6769	0.7013
CNN	512	tail	0.6203	0.7387	0.6743	0.7115

Table G.12: Performance scores for neural networks on GloVe-embeddings (dim 50)

G.4 Hugging Face Transformer Models

Model	Max Length	Precision	Recall	F_1	F_2
albert-base-v2	256	0.8884	0.8442	0.8657	0.8527
albert-base-v2	512	**0.9207**	0.8166	0.8655	0.8355
bert-base-uncased	256	0.8791	0.9029	0.8909	0.8981
bert-base-uncased	512	0.8750	0.8794	0.8772	0.8785
distilbert-base-uncased	256	0.8995	0.9097	0.9046	0.9077
distilbert-base-uncased	512	0.9075	0.8869	0.8971	0.8910
roberta-base	256	0.8925	**0.9368**	**0.9141**	**0.9276**
roberta-base	512	0.9114	0.9045	0.9079	0.9059

Table G.13: Performance scores for Large Language Models



NTNU

Norwegian University of
Science and Technology