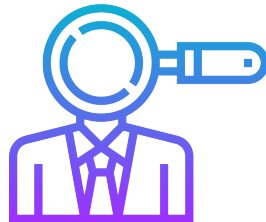


King Saud University
Collage of Computer and Information Sciences
Computer Science Department



“Personality Prediction Using Machine Learning Techniques”

CSC 497 – Final Report

Prepared by:

Khalid Alqahtani - 439101993

Fahad Almutairi - 439102401

Moath Alothman - 439102651

Abdulwhab Alhazmi - 438103416

Albaraa Aldohayan- 439102620

Supervised by:

Prof. Mohamed Maher Ben Ismail

Research project for the degree of Bachelor in Computer Science

First Semester 1444

Fall 2022

Acknowledgements

We would like to express our gratitude to the faculty members of King Saud University, and most importantly Prof. Mohamed Maher Ben Ismail for offering us the opportunity to conduct research under his supervision.

English Abstract

Personalities, as varied as they are, inform behavior in many ways, but their understanding is seldom harnessed to make decisions in the many spheres of life despite the useful insights they can bring forth. With the relatively recent popularization of personality psychology, however, there is growing momentum towards actualizing the potential of these insights. Today, we have personality tests that are empirically proven to predict behavior with the backing of mountains of research and data. Our goal in this paper is to make use of these untapped personality insights by intersecting Language Psychology with Natural Language Processing (NLP) and novel Machine Learning (ML) techniques that involve the use of state-of-the-art Deep Learning (DL) pre-trained models, also called Transformers—namely XLNet, BigBird, and ConvBERT, along with the appropriation of NLP techniques such as sentiment analysis and others—in the hopes of processing textual data and coming out with an elaborate classification of the author’s personality using The Big Five personality framework. This task can aid in a multitude of use cases, for which traditional survey-based personality tests are insufficient. Such use-cases are present in personalized marketing, job screening, recommender systems, and forensic linguistics for example. With text-based personality classification being relatively scarce, we are hoping to inspire and facilitate further research with our findings. To these ends, we have created a comparative study of various techniques and fine-tuned the three aforementioned models with model averaging to be more streamlined to textual personality prediction tasks. In particular, we have achieved a maximum F1 score of 86% in the trait Openness and an average of 69% across all 5 traits of The Big Five Personality framework. Furthermore, we have found the main personality dataset in the field, namely ”mypersonality”, to be lacking and imbalanced in many aspects.

Arabic Abstract

تقوم الشخصيات، على الرغم من تنوعها، بالتأثير على السلوك بعدة طرق، ولكن نادراً ما يتم تسخير فهمها لاتخاذ القرارات في العديد من مجالات الحياة على الرغم من الأفكار المفيدة التي يمكنها تقديمها. ومع ذلك، مع الانتشار النسبي لعلم نفس الشخصيات مؤخراً، هناك زخم متزايد نحو تحقيق إمكانات هذه الأفكار. لدينا اليوم اختبارات شخصية ثبت تجريبيًا أنها تتنبأ بالسلوك بدعم من جبال من البحث والبيانات. هدفنا في هذا البحث هو الاستفادة من هذه الاستنتاجات الشخصية غير المستغلة من خلال تقاطع علم نفس اللغة مع معالجة اللغة الطبيعية NLP وتقنيات التعلم الآلي ML الجديدة التي تتضمن استخدام أحدث تقنيات التعلم العميق DL والنماذج المدربة مسبقاً، وتسمى أيضًا Transformers — وهي XLNet و BigBird و ConvBERT جنباً إلى جنب مع اعتماد تقنيات البرمجة اللغوية العصبية مثل تحليل المشاعر وغيرها على أمل معالجة البيانات النصية والخروج بتصنيف مفصل لشخصية المؤلف باستخدام إطار ال Big Five. يمكن أن تساعد هذه المهمة في العديد من المواقف العملية، والتي لا تكفيها اختبارات الشخصية التقليدية القائمة على الاستبيان. المواقف العملية هذه موجودة في التسويق الشخصي، وفحص الوظائف، وأنظمة التوصية، واللغويات الجنائية على سبيل المثال. نظراً لندرة تصنيف الشخصيات إستناداً إلى النصوص نسبياً، نأمل في إلهام وتسهيل إجراء مزيد من الأبحاث استفادةً من النتائج التي توصلنا إليها. تحقيقاً لهذه الغايات، انشأنا دراسة مقارنة لمختلف التقنيات وضبطنا النماذج الثلاثة المذكورة أعلاه مع توسط نماذج التعلم العميق ليكون أكثر انسيابية لمهام التنبؤ بالشخصية بناءً على النصوص. على وجه الخصوص، لقد حققنا درجة F1 كحد أقصى تبلغ ٨٦ % في سمة الإنفتاح ومتوسط ٦٨% عبر جميع السمات الخمس لإطار السمات الخمس الكبرى للشخصية. علاوة على ذلك، وجدنا مجموعة البيانات الرئيسية في مجال الشخصيات بأسم "mypersonality"، غير كافية وغير متوازنة في العديد من الجوانب.

Table of Contents

1	Chapter 1: Introduction	1
1.1	Problem Statement	2
1.2	Goals and Objectives	2
1.3	Proposed Solution	2
1.4	Research Scope	2
2	Chapter 2: Background	3
2.1	Personality and Personality Models	3
2.1.1	Personality Models	3
2.1.1.1	Five-Factor Model	3
2.1.1.2	Myers Briggs Type Indicator	4
2.1.2	Personality Test	5
2.2	Natural Language Processing	6
2.2.1	A brief and general history of Natural Language Processing	6
2.2.2	Common NLP Tasks	7
2.2.2.1	Tokenization: Sentence Segmentation & Word Tokenization	7
2.2.2.2	Stop-Words	8
2.2.2.3	Text Normalization: Stemming & Lemmatization	8
2.2.2.4	Part of Speech Tagging	9
2.2.2.5	Named-Entity Recognition	10
2.2.2.6	Relation Extraction	11
2.2.2.7	Coreference Resolution	11
2.2.3	Text Representation	11
2.2.3.1	Bag of Words	12
2.2.3.2	Frequency Vector	12
2.2.3.3	One-Hot-Encoding	13
2.2.3.4	Why Raw Frequency Vectors Do Not Suffice	13
2.2.3.5	Term Frequency-Inverse Document Frequency (TF-IDF)	14
2.2.3.6	Term Frequency-Inverse Gravity Moment (TF-IGM)	15
2.2.3.7	Distributed Representations: Word Embeddings	15
2.3	Deep Learning	17
2.3.1	Architectures of Neural Networks	18
2.3.1.1	Convolutional Neural Network	18
2.3.1.2	Recurrent Neural Network	18
2.3.1.3	Long Short-Term Memory	19
2.3.1.4	Gated Recurrent Unit	19
2.3.1.5	Transformer Model	19
2.4	Supervised Learning	20
2.4.1	Classification	22
2.4.1.1	Binary Classification	22
2.4.1.2	Multi-Class Classification	23
2.4.1.3	Multi-Label Classification	24
2.4.2	Regression	24
2.5	Ensemble Learning	24
2.5.0.1	Bagging	24
2.5.0.2	Stacking	25
2.5.0.3	Boosting	26

3	Chapter 3: Literature Review	28
3.1	Machine Learning based solutions	28
3.2	Deep Learning based solutions	29
4	Chapter 4: Proposed Work	35
4.1	Preprocessing	35
4.2	Sentiment Analysis	36
4.3	NRC Emotion Lexicon	36
4.4	XLNet	36
4.5	BigBird	38
4.6	ConvBERT	38
4.7	Feed-Forward Neural Network	40
5	Chapter 5: Experimental Settings	41
5.1	Dataset	41
5.2	Performance Measure	41
5.2.1	Precision	41
5.2.2	Recall	41
5.2.3	Accuracy	41
5.2.4	F1 Score	41
5.3	K-Fold Cross Validation	43
6	Chapter 6: Experiments	44
6.1	Experiments' Environment	44
6.2	Experiment Issues	44
6.3	Data Exploration	45
7	Chapter 7: Results	47
7.1	Model Hyperparameters	47
7.1.1	Maximum Sequence Length	47
7.1.2	Number of Epochs	47
7.1.3	Batch Size	48
7.1.4	Learning Rate	49
7.1.5	Learning Rate Scheduler	50
7.1.6	Weight Decay	50
7.2	Unpreprocessed Dataset's Results	51
7.3	Applying Preprocessing	51
7.4	Applying Stemming	52
7.5	Applying Lemmatization	52
7.6	Applying Stop-words Removal	53
7.7	Applying NLP Features	53
7.7.1	Applying Sentiment Analysis	53
7.7.2	Applying The NRC Emotion Lexicon	54
7.7.3	Applying The NRC Emotion Lexicon With Sentiment Analysis	54
7.8	Applying Dataset Undersampling	55
7.9	Best Model Selection	56
8	Chapter 8: Conclusions and Future Works	58
9	References:	59

Table of Figures

1	Example Social and Emotional Skills Organized Under the Big Five Factors [26].	4
2	MyersBriggs Types [27].	5
3	Word tokenization based on the NLTK “sent_tokenize” module [113].	7
4	Word tokenization based on the NLTK “word_tokenize” module [113].	8
5	Using NLTK’s post_tag module for POS tagging.	9
6	Named-entity recognition using displaCy by explosion.ai [94].	10
7	Relation extraction using NLTK’s sem.relextract module [95].	11
8	Coreference resolution using huggingface’s neural coreference resolution system [100].	11
9	A co-occurrence matrix between words and documents. Rows are documents, and columns are words. Elements represent integer frequencies of the words [109]. . .	12
10	“Example of two encodings of the phrase ”red sticky clay”: numerical encoding and one-hot encoding.” [103].	13
11	“The term-document matrix for four words in four Shakespeare plays. The red boxes show that each word is represented as a row vector of length four.” [110].	14
12	Word embeddings in embedding space encoding the notion of analogy [104]. . .	16
13	A typical Neural Network [60].	17
14	The relationship between the network, layers, loss function, and optimizer [59].	18
15	CNN architecture [66].	18
16	A recurrent neural network [67].	19
17	LSTM and GRU architectures [64].	19
18	The Transformer - model architecture [65].	20
19	With the example of identifying fruits, a diagram depicting the process of supervised learning is shown [11].	21
20	Standard logistic function [10].	22
21	Bagging Ensemble [73].	24
22	Stacking Ensemble [75].	25
23	Boosting Ensemble [74].	26
24	The architecture proposed in [24].	30
25	A structural representation of the model (rd: recurrent dropout, l2: l2 regularizer) [35].	31
26	The architecture of the proposed system inspired by [44].	35
27	Permutation language modeling with predicting x3 as the task given the same input sequence x but with different factorization orders. The order is randomly permuted and each token’s prediction is based on its preceding tokens [48]. . .	37
28	(a) (b) (c) building blocks, (d) culminating in BigBird’s attention mechanism. White color denotes the absence of attention. (a) r = 2, (b) w = 3 (c) g = 2. (d) the combined BigBird model [128].	38
29	Illustration of self-attention, dynamic convolution and span-based dynamic convolution [129].	39
30	Illustration of mixed-attention block [129].	39
31	K-fold cross-validation with K = 5 [118].	43
32	Class imbalance in the dataset.	45
33	The relation between authors and the text in the dataset makes a right-skewed distribution.	46
34	Training and validation loss per epoch.	48
35	Training and validation loss per epoch.	48
36	Training loss per learning rate.	49

Table of Abbreviations

Abbreviation	Expanded-form/Stands for
ANN	Artificial Neural Network
BoW	Bag-of-Word
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
GPU	Graphics Processing Unit
GBR	Gradient Boosting Regression
KNN	K-Nearest Neighbors
LR	Logistic Regression
LSTM	Long Short-Term Memory
MBTI	Myers Briggs Type Indicator
ML	Machine Learning
MLP	Multi-Layer Perceptron
MNB	Multinomial Naive Bayes
NB	Naïve Bayes
NER	Named-Entity Recognition
NLP	Natural Language Processing
NLTK	Natural Language Tool Kit
PoST	Part-of-Speech Tagger
RNN	Recurrent Neural Network
SNA	Social Network Analysis
SVM	Support Vector Machine
TF-IDF	Term Frequency-Inverse Document Frequency
TF-IGM	Term Frequency-Inverse Gravity Moment
XGBoost	Extreme Gradient Boosting
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
ConvBERT	Convolutional Bidirectional Encoder Representations from Transformers
DistilBigBird	Distilled Big Bird
XLNet	Generalized Autoregressive Pretraining for Language Understanding

Chapter 1: Introduction

With the considerable variation in personalities, and with the abundance of textual data roaming the internet, traces of personality are sure to be found scattered between lines of text [1-4], only visible to the keen eyes of linguistic psychologists or those of their ilk. That statement, however, seems to only hold true in the past as we will later see the techniques of linguistic psychologists being replicated or, rather, paralleled using automated rule-based systems and statistical models [4-7]. The techniques in question fall under the field of author profiling. Historically, traditional implementations of author profiling techniques were limited to on-paper text and were carried out by trained linguistic psychologists. The work environment now is vastly different. We now have a preponderance of text due to the surge of the internet in the last few decades, along with unprecedented computing power capable of churning out textual insights on command.

"Author profiling" is a problem of growing momentum. It can be defined as the analysis of text to discover various aspects related to the author, using either a stylistic, content-based approach, or both [114]. Text carries information not just about its referents but also its referrers. In other words, text carries implicit linguistic markers that may reveal insights about its author, whether these insights be in the form of their psychological state, level of education, or personality as is the case with the insights our research focuses on [1-4]. This "personal" dependence, between author and text, is one of the core assumptions in language psychology, and one this research will also go on to assume.

The history of personality testing, which is predominantly survey-based, has been that of drudgerous manual labor, from compiling tests, requiring often unavailable input, and to having multiple interpretations by the subjects. Simply put, the traditional process of personality testing, although more direct, cannot fulfill the demands for swifter use-cases present in many fields. On the other hand, our research, as an attempt to tackle the author profiling problem, uses personality traits as a medium by which an author is correctly profiled, and, contrary to traditional manual methods, leverages known techniques in Machine Learning and Natural Language Processing, abbreviated as ML and NLP respectively, to reach maximal accuracies with reasonable efficiency. Although not intended to replace survey-based personality tests, these techniques fill the vacancies left by them.

Examples of the aforementioned fields abound: personalized marketing, language psychology, job application filtering, library cataloging, recommender systems, employee-to-job matching, and forensic linguistics, among others. In recommender systems, for example, some researchers have developed a model that recommends computer games to players according to their personality traits [2]. In personalized marketing, studies have shown that the personality of an individual has a strong impact on their purchasing choices and can work better than traditional correlates such as age and gender [3]. In language psychology, detecting personality traits automatically can assist in the discovery of more complex and delicate relationships between people's behaviorism and personality traits. This will aid in discovering new dynamics of the human psyche [4]. In job screening, personality traits affect one's suitability for certain jobs. For example, a company may want to hire a person who is comfortable speaking to large audiences. They can then narrow down the number of candidates to those high in extroversion, since it's a strong correlate, by inferring their personality through their written texts [5]. In forensic linguistics, a personality prediction system can be used for authorship attribution, which can greatly assist in cybercrime investigation [6]. In employee-to-job matching, a team

of researchers have found a considerable correlation between occupation and job suitability [8]. Although many of these fields have not actively integrated automated text-based personality prediction in them, these techniques have the transformative potential to revolutionize them.

In order to represent personalities, personality frameworks are required, of which The Big Five (OCEAN) and MBTI (Myers Briggs Type Indicator) are candidates. The Big Five framework describes personalities through 5 traits: Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism, each having a numbered range from 0 to 100. Given The Big Five’s continuous nature, as opposed to MBTI’s 16 discrete classes, Big Five will be the primary framework in this research.

1.1 Problem Statement

There is growing need for automated characterization of authors due to the impracticality of survey-based tests and their vulnerability to being manipulated, especially in work contexts. A promising solution can be found in Machine Learning (ML) and Natural Language Processing (NLP) models that take in text as input and output a personality prediction based on a personality trait framework such as The Big Five.

1.2 Goals and Objectives

The goal of this research is to construct a comparative study of preprocessing and model-building techniques in personality prediction.

The objectives of this research are:

1. Review the existing literature on personality prediction.
2. Build personality prediction models using different combinations of techniques (model-building or preprocessing).
3. Test the performance of the best performing combinations on the given datasets.

1.3 Proposed Solution

To answer to the aforementioned demands, text-based personality prediction systems are put forth as a viable solution. Pairing optimized models with the appropriate preprocessing steps is sure to give reasonable results. As such, we’re leveraging three state-of-the-art Transformers based on the classic Long short-term memory (LSTM) neural network. They are variants and offshoots of the popular Bidirectional Encoder Representations from Transformers (BERT). The three transformers in question are XLNet, BigBird, and ConvBERT. Each is different from the other and with its own merits and demerits. For the aim of making a better whole out of these 3, we propose the method of averaging their outputs for the final result. In conjunction with these transformers, NLP tasks will be carried out: sentiment analysis on the NRC Emotion Lexicon, and TF-IGM for obtaining term importance.

1.4 Research Scope

Our research focuses on the comparison of the various techniques utilized in text-based personality prediction systems from a statistical perspective. As such, peripheral considerations such as the distinction between stylistic and content-based textual analysis, or psycholinguistic model interpretations, etc..., are beyond the purview of our research. This research remains multidisciplinary however. Therefore, some concepts will be expounded upon as far as their intersection with Machine Learning is concerned.

Chapter 2: Background

We will cover some fundamentals that are useful to know before going forward. The topics covered will be explored briefly but adequately relative to the knowledge needed to understand the material. The main topics are Personality and Personality Models, Natural Language Processing, Deep Learning, Supervised Learning, and Ensemble Learning. Please note that this is not a comprehensive exploration of the material and shouldn't be expected to cover all bases.

2.1 Personality and Personality Models

Each person has their own personality that defines them. Explaining personality is very hard and complicated, even when it comes to psychology personality is a difficult topic to explain. The reason why it is very difficult to discuss or describe what a personality is because there are many perspectives to where a personality of one person is derived from. For example, it could be developed from within the person or the environment or both of these factors. Personality as stated by [14] “a pattern of relatively permanent traits and unique characteristics that give both consistency and individuality to a person's behavior”. However, traits evolve and change and appear differently in different people. A trait is a reflection of one's individuality and consistency in their behavior through time. Some traits can be related to a group of people, but each single person has their own set of unique traits. Every person is similar to one another but what makes someone different are their characteristics and qualities.

2.1.1 Personality Models

Our regular social interactions are guided by an instinctive understanding of personality. As an example, we predict that a family member will be a good artist as a result of their “creative” nature. Scientific models of personality give a structured strategy for defining, explaining, and predicting individual differences, whereas lay conceptualizations of personality are only loosely defined and often implicit. Rather than considering all the variables, scientific models of personality classify personality into traits. “Traits” refer to relatively consistent ways of thinking, behaving, and feeling across situations [15].

2.1.1.1 Five-Factor Model

The five-factor model (Big Five) was originally developed in 1949. The big 5 personality traits is a theory established by D. W. Fiske and later expanded upon by other researchers including Norman (1967), Smith (1967), Goldberg (1981), and McCrae Costa (1987). The big five model classifies personalities into five traits.



Figure 1: Example Social and Emotional Skills Organized Under the Big Five Factors [26].

These five traits can be described as: (i) Openness can be roughly defined as openness to experience who are willing to try the unknown associated with an attitude of being curious, creative, insightful, imaginative, and having a wide variety of interests [16]. (ii) Conscientiousness characterized by being mindful of details and reliable associated with an attitude of being goal-oriented, organized, and self-disciplined [16]. (iii) Extraversion is characterized by sociability and talkativeness, associated with an attitude of being active and forthcoming, and a desire for social relationships [16]. (iv) Agreeableness characterized by trust, kindness, and affection, and associated with an attitude of being friendly, warm, and sensitive toward others [16]. (v) Neuroticism can be characterized by sadness, moodiness, and emotional instability, and associated with an attitude of worrying and nervousness [16].

2.1.1.2 Myers Briggs Type Indicator

The Myers-Briggs Type Indicator or MBTI [115] is an extremely popular personality inventory which has received widespread use over the last years [116]. The MBTI is a self-report questionnaire designed to quantify non-psychopathological personality types as postulated in Jung’s psychodynamic type theory [115].

Each person’s four-letter code is used to indicate a person’s personality type, with a brief descriptive interpretation provided on the back of the report form. The MBTI appears to be an appropriate choice for a simple (four-dimensional), straightforward description of one’s personality make-up [17]. The instrument is given a score based on its own performance, and primarily utilized forced-choice (true/false) items. People are classified using four dichotomous dimensions either as extraverted (E) or introverted (I), sensing (S) or intuitive (N), thinking (T) or feeling (F), and judging (J) or perceiving (P) [17]. Combinations of the four preferences determine personality types. Each person is classified in terms of one of 16 possible four-letter codes (such as ISTJ, ISFP, ENFJ, and ESTP) [17]. Each type is said to represent a distinct set of behavioral tendencies that reflect differences in attitudes, viewpoints, and decision-making styles. The manual, test booklets, answer sheets, and score keys are all professionally produced materials.

What's Your Personality Type?

Use the questions on the outside of the chart to determine the four letters of your Myers-Briggs type. For each pair of letters, choose the side that seems most natural to you, even if you don't agree with every description.

<p>1. Are you outwardly or inwardly focused? If you:</p> <ul style="list-style-type: none"> • Could be described as talkative, outgoing • Like to be in a fast-paced environment • Tend to work out ideas with others, think out loud • Enjoy being the center of attention <p>then you prefer E Extraversion</p>	<ul style="list-style-type: none"> • Could be described as reserved, private • Prefer a slower pace with time for contemplation • Tend to think things through inside your head • Would rather observe than be the center of attention <p>then you prefer I Introversion</p>	<p>ISTJ Responsible, sincere, analytical, reserved, realistic, systematic. Hardworking and trustworthy with sound practical judgment.</p>	<p>ISFJ Warm, considerate, gentle, responsible, pragmatic, thorough. Devoted caretakers who enjoy being helpful to others.</p>	<p>INFP Idealistic, organized, insightful, dependable, compassionate, gentle. Seek harmony and cooperation, enjoy intellectual stimulation.</p>	<p>INTJ Innovative, independent, strategic, logical, reserved, insightful. Driven by their own original ideas to achieve improvements.</p>	<p>3. How do you prefer to make decisions? If you:</p> <ul style="list-style-type: none"> • Make decisions in an impersonal way, using logical reasoning • Value justice, fairness • Enjoy finding the flaws in an argument • Could be described as reasonable, level-headed <p>then you prefer T Thinking</p>	<ul style="list-style-type: none"> • Base your decisions on personal values and how your actions affect others • Value harmony, forgiveness • Like to please others and point out the best in people • Could be described as warm, empathetic <p>then you prefer F Feeling</p>
<p>2. How do you prefer to take in information? If you:</p> <ul style="list-style-type: none"> • Focus on the reality of how things are • Pay attention to concrete facts and details • Prefer ideas that have practical applications • Like to describe things in a specific, literal way <p>then you prefer S Sensing</p>	<ul style="list-style-type: none"> • Imagine the possibilities of how things could be • Notice the big picture, see how everything connects • Enjoy ideas and concepts for their own sake • Like to describe things in a figurative, poetic way <p>then you prefer N Intuition</p>	<p>ESTP Outgoing, realistic, action-oriented, curious, versatile, spontaneous. Pragmatic problem solvers and skillful negotiators.</p>	<p>ESFP Playful, enthusiastic, friendly, spontaneous, tactful, flexible. Have strong common sense, enjoy helping people in tangible ways.</p>	<p>ENFP Enthusiastic, creative, idealistic, perceptive, spontaneous, optimistic, supportive, playful. Value inspiration, enjoy starting new projects, see potential in others.</p>	<p>ENTP Inventive, enthusiastic, strategic, enterprising, inquisitive, versatile. Enjoy new ideas and challenges, value inspiration.</p>	<p>4. How do you prefer to live your outer life? If you:</p> <ul style="list-style-type: none"> • Prefer to have matters settled • Think rules and deadlines should be respected • Prefer to have detailed, step-by-step instructions • Make plans, want to know what you're getting into <p>then you prefer J Judging</p>	<ul style="list-style-type: none"> • Prefer to leave your options open • See rules and deadlines as flexible • Like to improvise and make things up as you go • Are spontaneous, enjoy surprises and new situations <p>then you prefer P Perceiving</p>
		<p>ESTJ Efficient, outgoing, analytical, systematic, dependable, realistic. Like to run the show and get things done in an orderly fashion.</p>	<p>ESFJ Friendly, outgoing, reliable, conscientious, organized, practical. Seek to be helpful and please others, enjoy being active and productive.</p>	<p>ENFJ Caring, enthusiastic, idealistic, organized, diplomatic, responsible. Skilled communicators who value connection with people.</p>	<p>ENTJ Strategic, logical, efficient, outgoing, ambitious, independent. Effective organizers of people and long-range planners.</p>		

Figure 2: MyersBriggs Types [27].

2.1.2 Personality Test

Finding out a person's personality can be done via a personality test. Finding the traits that people consistently display in a range of contexts can be done by evaluating and testing their personalities. Personality tests can aid in the clarification of a clinical diagnosis, the direction of therapeutic interventions, and the prediction of how people would react in various situations [25].

Types of Personality Tests

There are two basic types of personality tests:

- 1- Projective tests and self-report inventories: Test-takers read questions and then rate how well the question or statement applies to them in self-report inventories [18].
- 2- Projective tests entail presenting the test-taker with an ambiguous scene, object, or scenario and asking them to interpret the test item. The Rorschach Inkblot Test [19] is a well-known example of a projective test.

Uses of Personality Tests

Personality tests can be used for a variety of reasons, including:

- Diagnosing psychological problems.
- Looking at changes in personality.
- Screening job candidates [20].

Personality tests are occasionally employed in forensic contexts to conduct risk assessments, determine competency, and resolve child custody disputes [21]. Personality testing is used in a variety of settings, including school psychology, career and occupational counseling, relationship counseling, clinical psychology, and employment testing.

Potential Pitfalls

While personality tests might be beneficial at times, that does not mean they are not without disadvantages and potential hazards.

1- Deception Is Possible

One of the most significant drawbacks of self-report inventories is the potential for people to lie when answering questions. Even if dishonesty may be detected using procedures, people might nevertheless give misleading responses in order to "fake good" or appear more socially acceptable and desired [22].

2- Introspection Is Needed

Another issue is that people aren't always adept at reporting their own conduct accurately [23]. People tend to exaggerate certain features (especially those deemed socially desirable) while minimizing others. This can have a significant impact on a personality test's accuracy.

3- Results May Be Inconsistent

Personality tests aren't always accurate or valid. Validity is concerned with whether a test is truly measuring what it purports to measure [24]. Reliability relates to a test's consistency, whereas validity is concerned with whether the test is truly measuring what it claims to measure.

2.2 Natural Language Processing

Natural Language Processing is an umbrella term that comes about within the intersection of computer science, machine learning, and linguistics [84-85, 89]. It can be understood as covering "any kind of computer manipulation of natural language" from as simple as counting word frequencies in a document to the ambitious goal of having a "complete understanding" of human language [83-84].

2.2.1 A brief and general history of Natural Language Processing

Having generally started in the 1950s, NLP's earliest attempts were in the field of automated machine translation, in which text or speech are translated from one natural language to another. The first strands of NLP were that of rule-based frameworks/techniques such as context-free grammars (CFG), regular expressions (RegEx), and tokenizers. And while inadequate in tackling the more ambitious NLP tasks, they are still employed in modern NLP as preprocessors of text before having the text fed to statistical machine learning models, whether shallow or deep [86]. The aforementioned statistical models are what followed the rule-based era of NLP [86-87], as the 1980s saw a fundamental "reorientation" of the methods used. The methods became more statistically driven and rules, for example, were associated with probabilities predicted by machine learning models that were trained on text corpora [86]. NLP practitioners looked less for deep analyses of the text using linguistic rules and more for simple generalizable approximations [86-87]. Evaluations however became more rigorous as they were drawn from the annotated data (data with labels) that the language machine learning models were trained on [86-87]. It must be noted that in the end, rule-based and statistical approaches are applied in tandem and their relationship is more complementary than incompatible [86, 88].

2.2.2 Common NLP Tasks

The modern NLP systems, of which the machine learning models are usually the prime movers, require preprocessing to be made on the text that is inputted into them. We will briefly go over a number of common NLP tasks that serve as text preprocessors or as end goals themselves. Note that we are assuming the natural language in question to be English, therefore the additional challenges present in other natural languages won't be addressed.

2.2.2.1 Tokenization: Sentence Segmentation & Word Tokenization

Sentence Segmentation

Typically the first preprocessing step done, sentence segmentation is the breaking up of text into separate sentences. In English, sentences generally end with periods, exclamation marks, or question marks, but some edge cases could pose a problem, such as abbreviations and forms of addresses (U.S.A., Dr., Mr., etc.), or ellipses (...). These cases have been addressed by many NLP libraries, of which the Natural Language Tool Kit (NLTK) is a common one. Figure 3 shows a body of text being inputted into a sentence segmenter and the resulting output as per NLTK's "sent_tokenize" module [90-91].

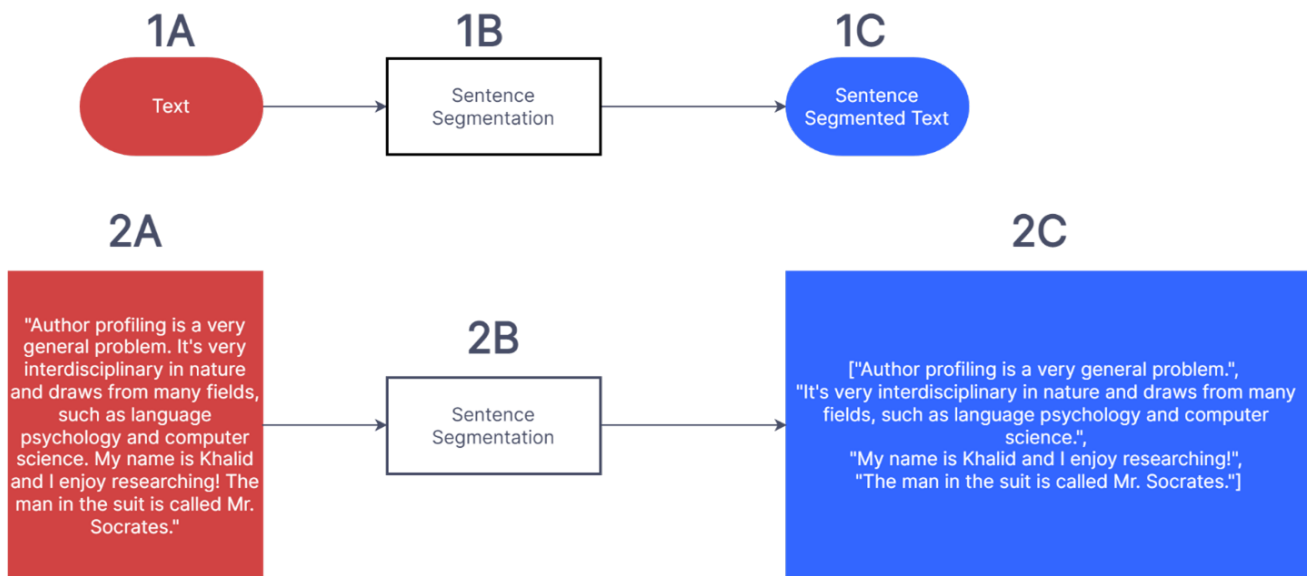


Figure 3: Word tokenization based on the NLTK "sent_tokenize" module [113].

Word Tokenization

Similar to sentence segmentation but here the segmentation is based on words, not sentences. Edge cases also appear in word tokenizers (depending on the rules specified) and must be carefully considered. Consider the following sentence: "Shaquille O'Neal is a great basketball player standing tall at 2.16 m". When using NLTK, O'Neal gets tokenized into 3 separate words O, ', Neal instead of 1 as simply O'Neal. To address such problems, a custom tokenizer with carefully specified rules might be necessary. Fig. 5 shows a sentence being inputted into a word tokenizer and the resulting output as per NLTK's "word_tokenize" module [90-91].

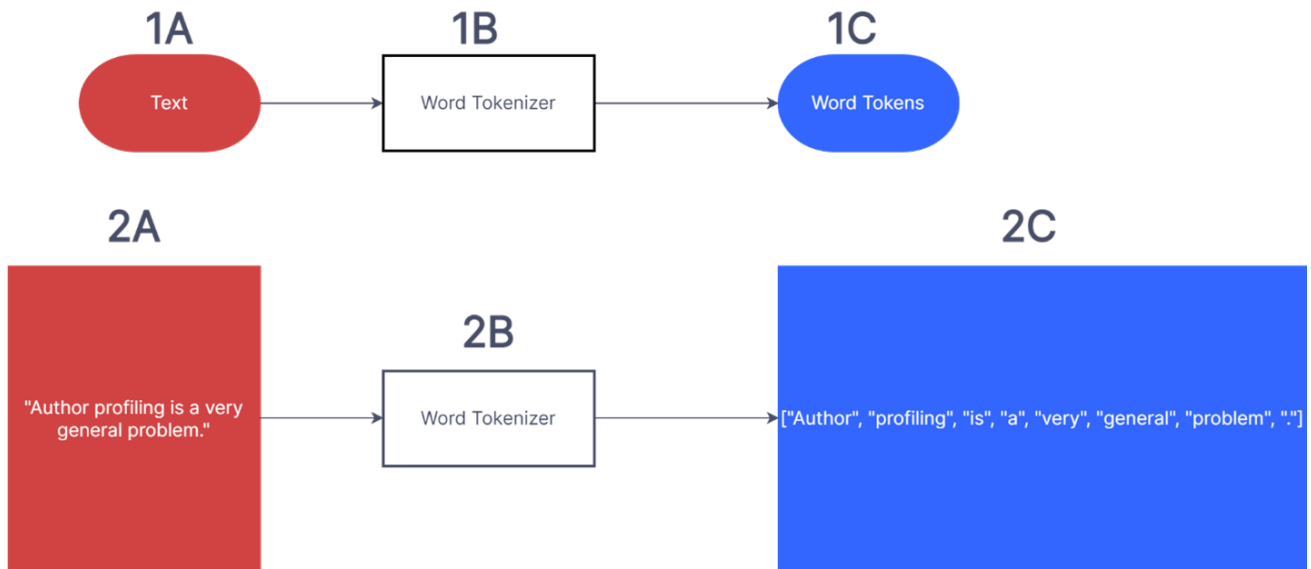


Figure 4: Word tokenization based on the NLTK “word.tokenize” module [113].

2.2.2.2 Stop-Words

When dealing with large text corpora, it’s easy to see that some words are better predictors than others in a specified task. In fact, some words carry little insight, act as noise, don’t explain the variance between data-points well, and add meaningless dimensionality to the data. Such words are termed “stop-words” and it’s common practice to remove them in NLP pipelines. The corpora of “stop-words” vary depending on the task. However, there exist common corpora that fit the use-cases of many ordinary tasks where lexical meaning is most important. One of the aforementioned corpora, in the case of NLTK’s stop words, includes words such as “a”, “the”, and “an” as they don’t provide insight for typical meaning-driven tasks. If the NLP task at hand were that of grammar correction, then such stop-words would not be removed as they are good predictors of correct grammar. It’s important to think of the nature of the task at hand before going with a specific corpus. A custom-made stop-word corpus could be useful in such cases [84-85, 90-91].

2.2.2.3 Text Normalization: Stemming & Lemmatization

The multiplicity of representations for each token can pose a problem in text processing. Take the tokens “Mr.” and “Mister” for example, they are essentially two separate representations of the same thing. In a case where only meaning matters, this adds dimensionality and creates an unnecessary separation between the two tokens, possibly losing out on meaningful information. A way to fix this problem is to simply change “Mister” to “Mr.” or vice-versa, merging them into a singular word token. Other examples are “two” and “2”, “Khalid” and “Khaled”, “COMPUTER” and “computer”, “Hello” and “Hello” (misspelling), etc. These are relatively simple examples of text normalization. Moving up in complexity, a token such as “better” could be reduced to “good” using lemmatization, a technique we’ll discuss below. Although, such a reduction clearly is unhelpful in a case where the comparative nature of the term needs to be preserved, as in the sentence “Product A is better than B”. Similar to the task of designating a corpus of stop-words for use, having the optimal mapping between tokens is dependent on the use-case at hand [84, 90].

Stemming

Stemming is a text normalization technique wherein the affixes to the words are removed, regardless if the resulting forms exist in a specified dictionary or not. An example would be "changed", "changes", "changer", "changing" all being reduced to the form of "chang" when inputted into some stemmer with specific rules. Here the word "chang" is the incorrect base form of the words, and that's where the difference between stemmers and lemmatizers lies as we will further explain below. Stemming is used commonly in user queries in search engines and in cases where one needs to reduce the dimensionality of the data [84, 86, 90].

Lemmatization

Lemmatization is a text normalization technique wherein the affixes to the words are removed only if the resulting forms exist in a specified dictionary. For example, the words "has" and "had" are reduced to "have" upon lemmatization, as "have" is the linguistically correct base form. Unlike stemming, lemmatization requires linguistic knowledge and has the extra step of dictionary-checking, making it slower in practice. Developing efficient lemmatizers is considered an open problem in NLP research [84, 86, 90].

2.2.2.4 Part of Speech Tagging

In many situations, it can be helpful to filter words based on the part of speech category they belong to (noun, verb, prepositions, or even present tense verbs, etc...). This is where a part-of-speech tagger (POST or POS-tagger) comes in. POSTs take a stream of tokens as input and map part of speech tags to each token based on its definition and or context (depending on the complexity of the POST in question). Figure 5 shows the result of a POST that has been inputted a tokenized sentence per NLTK's `post_tag` module [84, 86, 90].

```
Sentence before parts-of-speech tagging:
I love text-based author-profiling

Sentence after parts-of-speech tagging:
[('I', 'Personal pronoun (PRP)'),
 ('love', 'Verb, non-3rd person singular present (VBP)'),
 ('text-based', 'Adjective (JJ)'),
 ('author-profiling', 'Noun, singular or mass (NN)')]
```

Figure 5: Using NLTK's `post_tag` module for POS tagging.

2.2.2.5 Named-Entity Recognition

What if you want to filter specific words or phrases based on the type of entities they are? You use a named-entity recognition (NER) system. Much like POST, NER systems take a stream of tokens as input and map named entities to each token based on its definition and or context (also dependent on the complexity of the NER system in question). To better clarify, the token "Khalid" is a named entity of type "PERSON", whereas the token "about a week ago" is a named entity of type "DATE". Figure 6 shows the result of inputting text into spaCy's open-source visualizer (displaCy Named Entity Visualizer by explosion.ai) [84-86, 90, 93, 94].

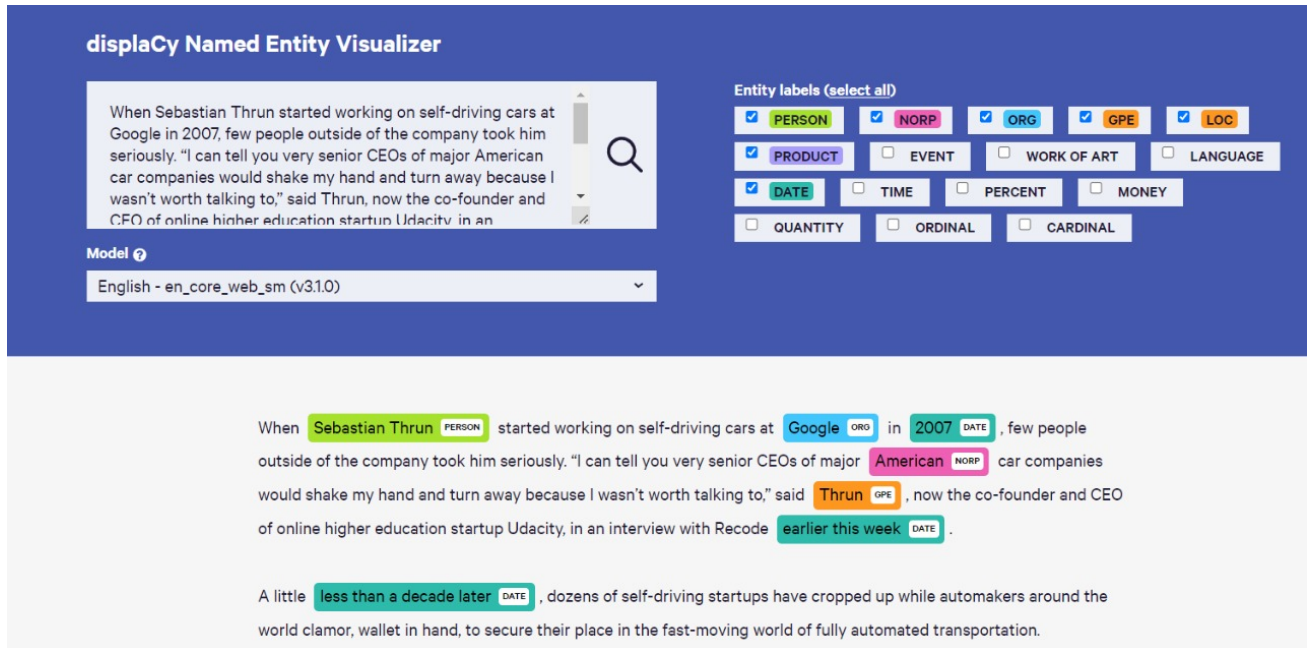


Figure 6: Named-entity recognition using displaCy by explosion.ai [94].

2.2.2.6 Relation Extraction

Often it's not enough to merely have unbound entities in a text, and relations between entities need to be automatically recognized. Take for example the sentence "Khalid is a student in King Saud's University", here a NER system can recognize that "Khalid" belongs to the entity type of "PERSON" and that "King Saud's University" belongs to the entity type of "ORG". But, that doesn't suffice for many use-cases, which is why NER systems are often paired with a relation extraction system that automatically finds that "Khalid" and "King Saud's University" are bound by the relation "is a student in", for example. Relations can be formalized when modeled in the form of triples (X, a, Y) where "X" and "Y" are separate entities that are bound by a relation "a" []. The previous example in formal terms is ([PER: "Khalid"], "is a student in", [ORG: "King Saud's University"])). Figure 7 shows more examples in the form of triples from NLTK's sem.relextract module [84, 90, 95].

```
[PER: 'Kivutha Kibwana'] ', of the' [ORG: 'National Convention Assembly']  
[PER: 'Boban Boskovic'] ', chief executive of the' [ORG: 'Plastika']  
[PER: 'Annan'] ', the first sub-Saharan African to head the' [ORG: 'United Nations']  
[PER: 'Kiriyyenko'] 'became a foreman at the' [ORG: 'Krasnoye Sormovo']  
[PER: 'Annan'] ', the first sub-Saharan African to head the' [ORG: 'United Nations']  
[PER: 'Mike Godwin'] ', chief counsel for the' [ORG: 'Electronic Frontier Foundation']
```

Figure 7: Relation extraction using NLTK's sem.relextract module [95].

2.2.2.7 Coreference Resolution

Coreference resolution is the task of identifying which mentions refer to which entities. For example, in the sentence, "Khalid likes learning about Natural Language Processing. He really likes it.", the mention "He" clearly refers to the entity "Khalid". Although, whether "it" refers to "Natural Language Processing" or the act of learning about it can be a hard problem even for humans. Figure 8 shows a sentence being inputted into huggingface's neural coreference resolution system based on neural networks and spaCy [96-100].

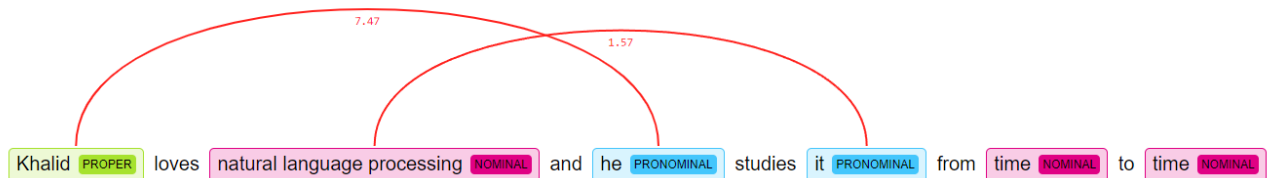


Figure 8: Coreference resolution using huggingface's neural coreference resolution system [100].

2.2.3 Text Representation

Text needs a representation if it is meant to be analyzed, manipulated, and fed into machine learning models. The standard representation does not suffice because it's unintelligible to computers and statistical/mathematical computations. Machine learning models

expect data in the form of 2D arrays with rows representing "instances" and columns representing "features". In this textual framework, "instances" can be words or entire documents of varying lengths, "features" as attributes of said instances that describe the words/documents along certain dimensions (columns). Examples of features are meta-data attributes such as "document length", "source", "publication date", etc. By representing text numerically through the generation of features, we're engaging in the process of what is called "feature extraction" or "vectorization". The resultant numerical encodings of the text are in the form of points in a high-dimensional space, a "semantic" space if you will, with their distances from each other representing similarity [102]. This transformation makes text workable and better amenable to mathematical operations. Below are some text representations used in the field [83, 101-102, 104].

2.2.3.1 Bag of Words


A very simple encoding upon which more complex encodings will be built is the Bag-of-Words (BoW) model. This encoding is done by making each word in the corpus's vocabulary its own feature with each element representing a relevant numerical value associated with the word (feature) within a document (instance). The "relevant numerical value" can represent many things: the frequency of the occurrence of the word, a boolean value representing whether the word occurred at least once or not (one-hot encoding), a weighted frequency such as "term frequency-inverse document frequency" (TF-IDF) [105, 112] or "term frequency inverse gravity moment" (TF-IGM) [106], etc., both of which build upon the bare-bones frequency counter mentioned first, and distributed representations (word embeddings) such as Word2vec [107] and GloVe [108]. Note that many cases use words as both instances and features as they look to capture co-occurrences between words, so it's not limited to a word-to-document representation [102].

2.2.3.2 Frequency Vector

This encoding is done by making each word in the corpus's vocabulary its own feature with each element representing the frequency of a word's (feature's) occurrence within a document (instance). This is a standard frequency counter with no weights assigned. Figure 9 shows two documents having frequency vectors [102, 109].

D1: He is a lazy boy. She is also lazy.

D2: Neeraj is a lazy person.



	He	She	lazy	boy	Neeraj	person
D1	1	1	2	1	0	0
D2	0	0	1	0	1	1

Figure 9: A co-occurrence matrix between words and documents. Rows are documents, and columns are words. Elements represent integer frequencies of the words [109].

2.2.3.3 One-Hot-Encoding

As used in text-based tasks, one-hot is a boolean vector encoding that assigns a value of 1 to a vector index if and only if a specified token exists in the document, and 0 otherwise. As you can imagine, this increases the dimensionality of the dataset and is thus advised against if you have a large dataset since the length of the one-hot vector would equal the number of tokens in the vocabulary. As a simple example, given the vocabulary "Khalid", "Moath", "Abdulwahab", the one-hot encoded vector would look like this when given these 5 documents:

Khalid = [1, 0, 0]

Abdulwahab = [0, 1, 0]

Moath = [0, 0, 1]

Moath Abdulwahab = [0, 1, 1]

Moath Abdulwahab Khalid = [1, 1, 1]

As you can see, each feature in the vector reflects the existence or lack thereof of a token in the document. Figure 10 better illustrates one-hot encoding in juxtaposition to standard numerical encoding [83, 101-102, 103-104].

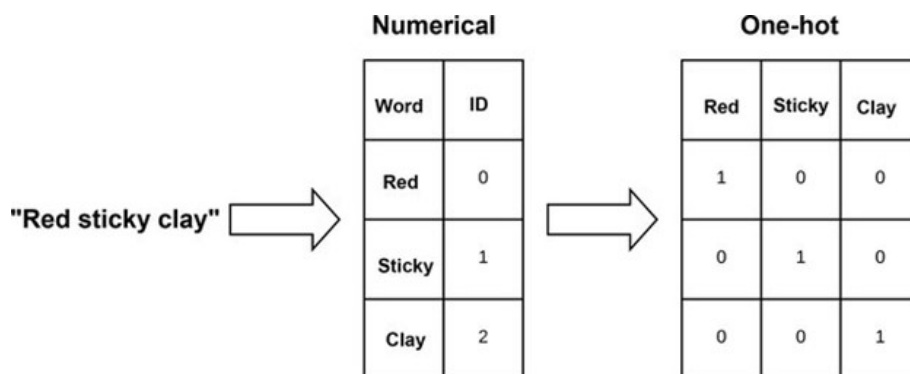


Figure 10: "Example of two encodings of the phrase "red sticky clay": numerical encoding and one-hot encoding." [103].

2.2.3.4 Why Raw Frequency Vectors Do Not Suffice

The problem with raw frequency vectors is that they are not discriminative enough to be informative for many use-cases. Simple extensions can be applied to them that capture better the notions of context sharing. To illustrate this with an example, some terms are too ubiquitous that they always share contexts with virtually every other term. Disregarding the obvious stop-words such as "a" as an example which can simply be removed, consider the term "good" which appears in many contexts despite its recognition being uninformative in a sense that is relevant to many use-cases, such as in the example shown by Figure 11, where the word "good" is not being adequately discriminative within the collection of Shakespeare plays due to its ubiquity [102, 110].

The authors of the book "Speech and Language Processing" [110] described as a paradox the fact that frequently occurring-or cooccurring-words are more important than those with lesser frequencies, yet if they occur too much such as "good", they are unimportant. To satisfy these constraints, many weightings and techniques were adopted, one of which is the Term Frequency-Inverse Document Frequency (TF-IDF) which we'll explain below.

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	14	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Figure 11: “The term-document matrix for four words in four Shakespeare plays. The red boxes show that each word is represented as a row vector of length four.” [110].

2.2.3.5 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is a frequency vector that can be broken down as having two parts: “TF” and IDF”. The “TF” part accounts for term frequencies within the document they appear in, with frequency in the raw count form, one-hot encoded, or divided by all terms in the document, etc. The “IDF” part accounts for term frequencies in relation to all of the documents in the corpus, thus giving more weight to rare terms with respect not only to the document but also to the entire corpus. In mathematical notation [110], let’s denote documents with “ d ”, and the collection of documents as “ D ”, with the number of documents in the corpus as “ N ”. Terms are denoted by “ t ”, the collection of terms within a document as “ T_n ”, and the number of documents “ d ” in which a term “ t ” occurs as “ df_t ” [102, 110, 105, 112].

You can count term frequency by considering “ T_n ” in order to better represent the prevalence of a specified term by dividing its frequency by the number of terms “ t ” in the document or even by that of the rest of the terms’ frequencies.

TF using the raw count can be given by:

$$tf_{t,d} = count(t, d) \quad (1)$$

whereas IDF can be given by:

$$idf_t = \frac{N}{df_t} \quad (2)$$

Commonly, however, the resultant values of the above forms of TF and IDF are scaled logarithmically before the product TF-IDF is applied and are respectively given by:

$$tf_{t,d} = \log_{10}(count(t, d) + 1) \quad (3)$$

$$idf_t = \log_{10}\left(\frac{N}{df_t}\right) \quad (4)$$

which when combined through a product operation gives us TF-IDF:

$$TF-IDF = tf_{t,d} \times idf_t \quad (5)$$

The lower the " df_t ", the higher the IDF. The lowest IDF possible is 1 and is only assigned to terms that appear in all of the documents within a corpus, and in turn, the lowest TF possible is 0. The way we interpret the resultant TF-IDF value for a given term is that the closer it is to 1, the more informative it is, and vice versa with respect to 0 [102, 110].

2.2.3.6 Term Frequency-Inverse Gravity Moment (TF-IGM)

TF-IGM (Term Frequency - Inverse Gravity Moment) is a term weighting scheme used to properly assess the weight of each class, incorporating a new statistical model [119]. The weight states how many words contribute to the class of documents. This method is suitable for use in personality prediction due to the fact it allows for a person to have more than one personality because it's able to separate label classes in textual data, especially for data that has more than one label. To calculate TF-IGM you first need to calculate TF and IGM using the following formulas:

$$TF = \frac{\text{Total appearance of a word in a document}}{\text{Total words in a document}} \quad (6)$$

$$IGM = 1 + \lambda \left(\frac{\text{Total appearance of a word in a document}}{\text{Total appearance of a word in each class}} \right) \quad (7)$$

In the weight of a term, λ represents an adjustable coefficient that is used to maintain a balance between global and local variables.

After calculating TF and IGM, now you can calculate TF-IGM by using the following formula:

$$TF - IGM = TF * IGM \quad (8)$$

TF-IGM value ranges from 0 to 1. A TF-IGM value will be assigned to each word in a document, and the words will be sorted by the highest value. High-value words will be used as a feature in classification models since it is assumed that these words contain the relevant meaning of a text with a certain class label.

2.2.3.7 Distributed Representations: Word Embeddings

Another way to encode textual data with is through the power of word embeddings. We have seen the previous representations each having their own merits, such as the case of TF-IDF where it encodes a notion of a word's "importance" within it. Here, when using word embeddings what is encoded is instead "similarity" in a way that is even more amenable to mathematical operations and can represent to a degree notions of analogy and opposition, and plot words as vectors in a vector space in a way that coincides with intuition. Word embeddings are trainable dense vectors, as opposed to sparse one-hot encodings, that represent aspects of given words through their values. A very famous example that illustrates the flexibility of word embeddings is the King-Queen analogy: if you have the words "King", "Man" and "Queen", and compute "King" - "Man" +

"Woman", the resulting value will be most similar to the word "Queen" as shown in Figure 12 [102, 104, 110].

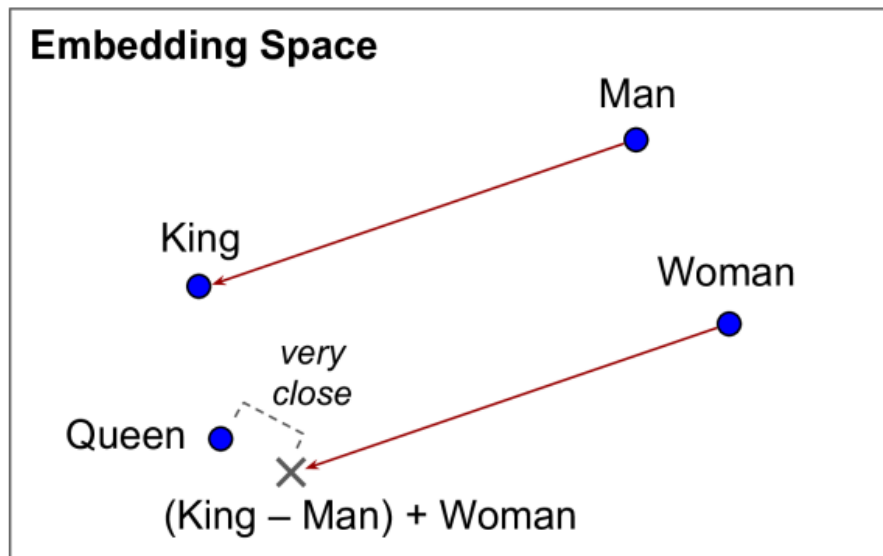


Figure 12: Word embeddings in embedding space encoding the notion of analogy [104].

2.3 Deep Learning

Deep learning is a subfield of machine learning that is inspired by the way the human brain functions [59], and it employs a structure known as a neural network, also known as an artificial neural network (ANN).

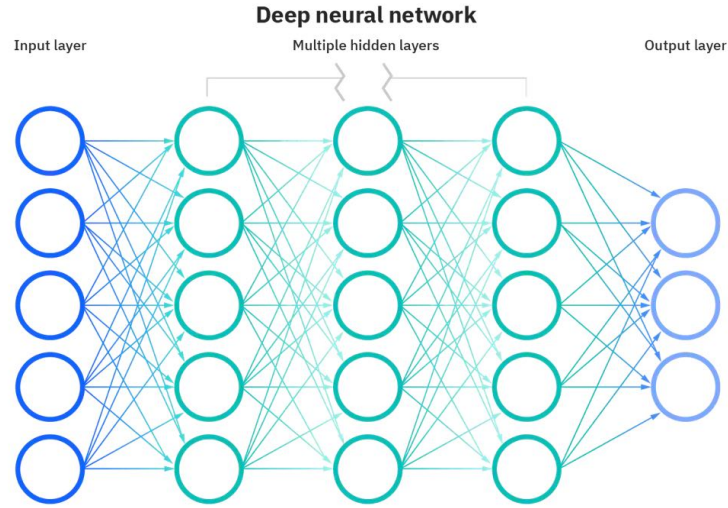


Figure 13: A typical Neural Network [60].

A neural network is made up of layers, each of which contains neurons. A typical neural network has three layers: an input layer, one or more hidden layers, and an output layer. Each neuron is connected to other neurons and has its own weight and threshold. If the neuron's output exceeds its associated threshold, the neuron will be activated and the data will be passed to the next layer; otherwise, the data will not be transmitted. Weights are used to show how important a variable is and how much it contributes to the outcome when it compared to other variables. Activation functions are used to determine whether or not a neuron's output should be forwarded to the next layer of the network.

The success of prediction is measured using a loss function, and the higher it gets, the worse our model performs. To reduce the loss function and enhance accuracy, optimizers are used to update the weights in the neural network [59].

The best values for the weights in all layers that contribute to finding the best mapping from input to output are found in a neural network by utilizing a loss function to compute the difference between the predicted value and true value, then using an optimizer to update the weights in the network [59]. Deep Learning saves us time by automating the feature engineering process, rather than having to do it manually as with machine learning algorithms [59].

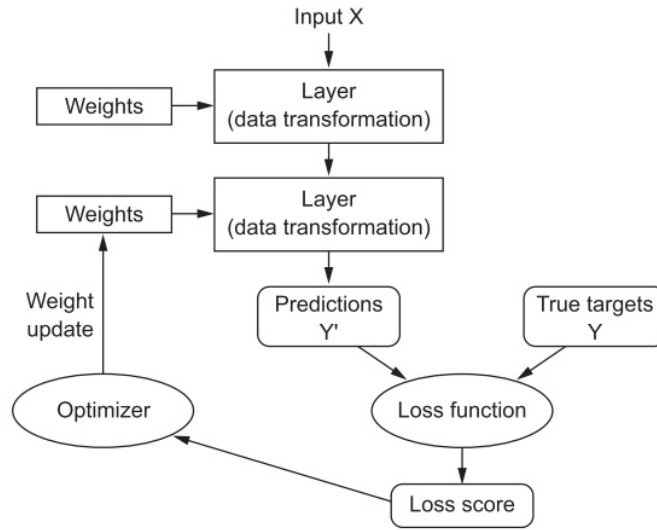


Figure 14: The relationship between the network, layers, loss function, and optimizer [59].

2.3.1 Architectures of Neural Networks

2.3.1.1 Convolutional Neural Network

This architecture was first introduced in this paper [58] by Yann LeCun. The different layers that make up CNN set it apart from other neural network architectures, as well as its greater performance with picture, audio, and speech input. The convolutional layer is the initial layer of a CNN, followed by successive convolutional layers or pooling layers, and finally the fully connected layer. CNN focuses on small features in the first layers, then increasingly recognizes larger features of the object until it recognizes the entire intended object.

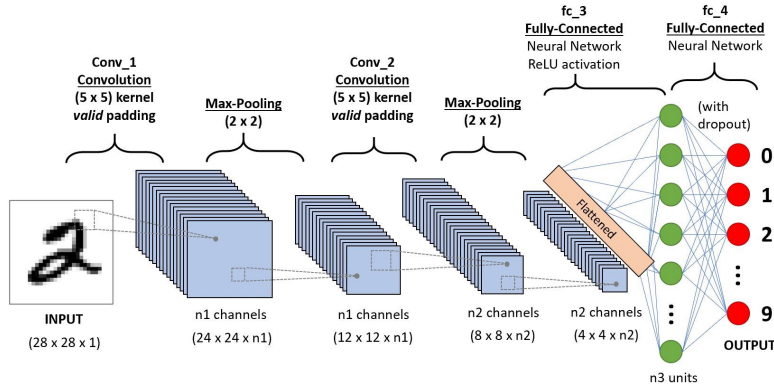


Figure 15: CNN architecture [66].

2.3.1.2 Recurrent Neural Network

This architecture was first described in [61], and it's specialized in processing sequential data to classify or generate new sequences. RNNs process sequences by iterating through the sequence elements and storing information about what it has learned so far to use it later to influence later output. As you can see in the image below, each neuron of the RNN has an internal loop. Problems related to text and speech processing can be solved with RNN.

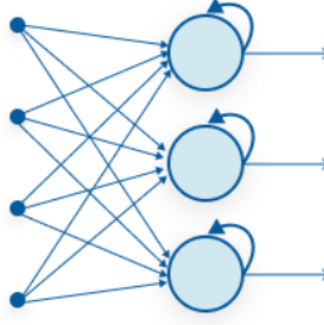


Figure 16: A recurrent neural network [67].

The issue with the basic RNN architecture is that if the present state's output is dependent on a far in the past state's output, the RNN model will be unable to generate an accurate prediction to the current state [59]. LSTM and GRU algorithms are designed to solve this issue.

2.3.1.3 Long Short-Term Memory

Hochreiter and Schmidhuber developed a variation of a RNN architecture which is called LSTM algorithm to solve the vanishing gradient problem [62]. By incorporating cells to the hidden layers of a RNN architecture. There are three gates in these cells: an input gate, output gate, and a forget gate. The information flow is controlled by these gates, which helps in predicting the output.

2.3.1.4 Gated Recurrent Unit

This work [63] proposes GRU, a new algorithm for solving the vanishing gradient problem. Its architecture is quite similar to that of LSTM as depicted in Figure 17.

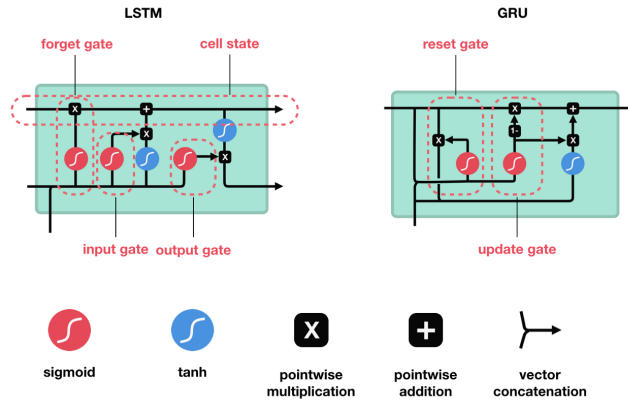


Figure 17: LSTM and GRU architectures [64].

2.3.1.5 Transformer Model

The study in [65] introduced the Transformer model. RNN, LSTM, and GRU are very useful, but they have some drawbacks, such as the fact that they process their input sequentially, which wastes a lot of time and memory. In contrast, transformers process their input in parallel using Graphics Processing Unit (GPU) and address the memory issue with self-attention, as well as adding a positional encoding to the input.

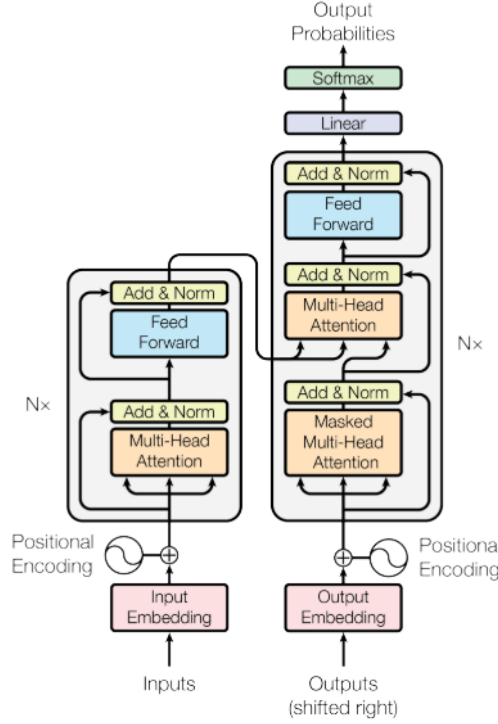


Figure 18: The Transformer - model architecture [65].

A transformer model’s architecture is made up of an encoder and a decoder, both of which are made up of $N \times$ layers, as shown in Figure 18.

2.4 Supervised Learning

Supervised learning is a subcategory of machine learning and we can define it as “a machine learning task to learn a function that maps inputs to outputs based on examples of input-output pairs” [7]. In this type of learning, there are 2 essential variables in a data-set: Features, also called inputs, and Labels, also called outputs. You will deal with pre-defined information, meaning that each data set that is stored will have a name that identifies it. In order to make the concept clear to us, let us assume that someone was wandering in an area and saw a new mineral. Before he recognized the name of the mineral, he knew what it is based on its features such as its hardness, luster, color, etc., and then a geologist supervised this process by giving these features a name or label so that the man could then associate the features with this label, and in this example the geologist is the supervisor of the process learning by giving the appropriate name for these features found in that mineral.

In mathematical terms, as mentioned above there are 2 essential variables in a data-set: Features (inputs), and Labels (output), Mathematically, the features can be expressed as x_j , where j starts at 1 and ends at “f” (last feature). No such expression for labels exists because there can only be 1 label for 1 observation. An observation is denoted by x_i , where i starts from 1 and ends at “n” (number of examples). Each x_i is basically a feature vector whose dimensionality equals the number of features, and whose elements start from x_1 and end at x_f . The labels for each respective observation can be expressed as y_i , where j starts at 1 and ends at “n” (number of examples).

The way this is used in action is when given an observation (feature vector), the supervised learning algorithm outputs a label consistent with the set of possible labels in the dataset. For example, if the input is ["150cm", "40kg", "20cm", "3kg", "1999", "red", "2", "4 cylinders"], the output could be "Mitsubishi Mirage".

How Supervised Learning Works

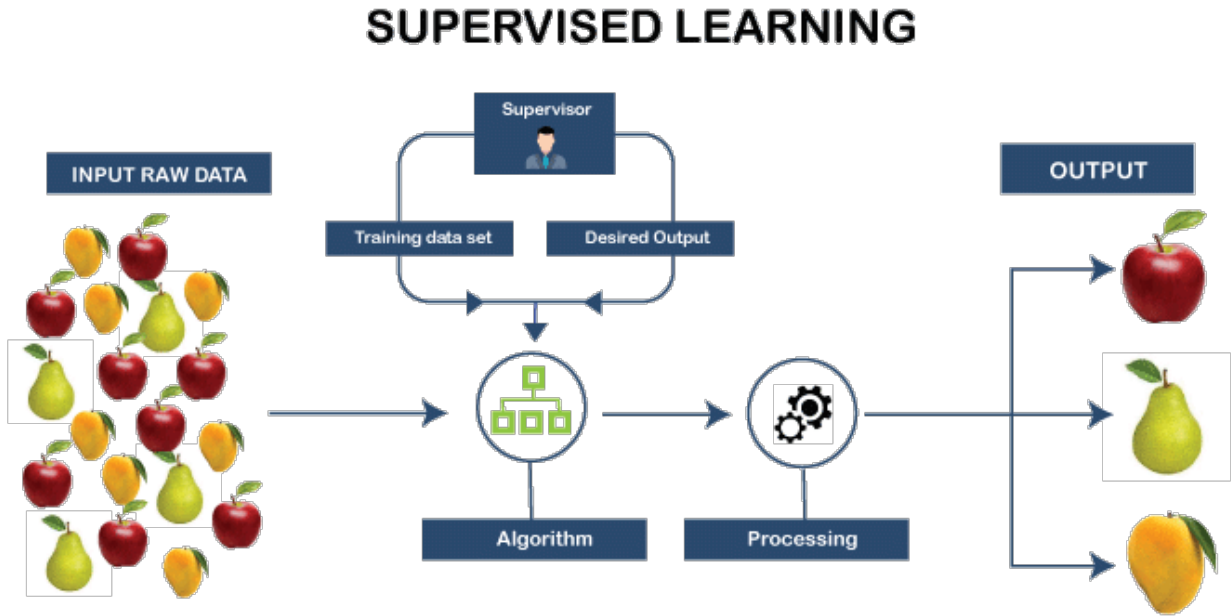


Figure 19: With the example of identifying fruits, a diagram depicting the process of supervised learning is shown [11].

Taking a Support Vector Machine (SVM) model that separates spam from non-spam as an example: $w x - b = 0$. This equation represents the hyperplane (decision boundary). Weights are denoted by “w” and Bias is denoted by “b”. Each “x” is a feature vector. $w x_i - b \geq +1$ if $y_i = +1$ (Spam), $w x_i - b \leq 1$ if $y_i = 1$ (Non-spam). This represents how the feature vectors are classified in the end. Predicting comes in the following form:

$$f(x) = \text{sign}(w^* x - b^*)$$

Why the Model Works on New Data

As in the case of the aforementioned SVM model, the model works on new data by associating the new data’s features with the data it has already seen (trained data). As long as the training data is good and has been correctly fed to the learning algorithm, it’s likely new data will follow suit in being appropriately classified.

Can we apply this concept to examples in our real world?

Yes, there are many applications to supervised learning, the most important of which are: text classification, face detection, signature recognition, and weather forecasting [8].

There are two types of supervised learning techniques: Regression and Classification.

2.4.1 Classification

Classification is a supervised learning approach that is a problem to classify a group of items into multiple classes [10]. There are many examples in the real life of uses of classification for example, when we predict students' grades in the form of a classification like $A+$, A , $B+$, B , *etc.* Also when we determine a person's blood type e.g. $O+$, $O-$, AB , B , *etc.* There are three main types of classification: Binary Classification, Multi-Class Classification, and Multi-Label Classification [10].

2.4.1.1 Binary Classification

Binary classification is the problem of classifying the members of a group into two groups [10]. There are many examples related to Binary classification, and most of them consist of answers in the form of "having" or "not having". For example, does the person have Covid-19? The answer would be positive or negative, another example would be if we were to rate the student as "passed" or "failed" and etc.

The most used algorithms in binary classification: Logistic Regression, k-Nearest Neighbors, Decision Trees, SVM, and Naive Bayes [12]. There are some algorithms that are specifically designed for Binary classification only and cannot be used in others, for example, Logistic regression and SVM.

Logistic Regression is a classification algorithm, not a regression. The word regression comes from the fact that the algorithm resembles a linear regression algorithm [10].

In logistic regression, we are trying to predict a variable Y from a set of values X . Y is a set of discrete variable classes, whereas X is the feature vector. We also optimize an equation to get the best possible weights that model the actual relationship between X and Y . The range of our model is $(0,1)$, where a prediction close to 0 means it belongs to the first class and a prediction of 1 means it belongs to the second class [10].

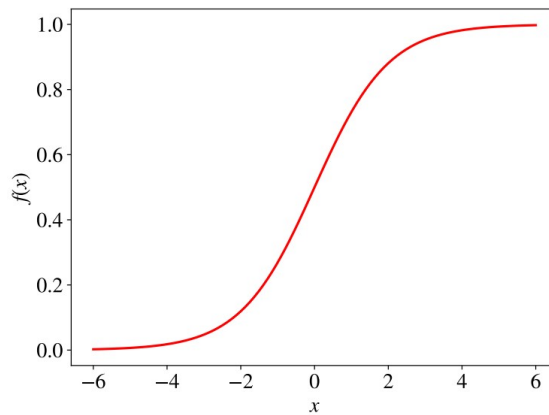


Figure 20: Standard logistic function [10].

Sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}},$$

As shown in Figure 20, we utilize the Sigmoid function to limit the range to $(0, 1)$. Where e is a mathematical constant called Euler's number. Figure 20 depicts its graph.

The logistic regression model is as follows:

$$f_{\mathbf{w},b}(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{1 + e^{-(\mathbf{w}\mathbf{x}+b)}}. \quad (9)$$

The expression $\mathbf{w}\mathbf{x}+b$ is derived from linear regression, where \mathbf{w} is a D-dimensional vector of parameters, and b is a real value. The notation $f_{\mathbf{w},b}$ denotes that the model f has two parameters: \mathbf{w} and b .

We can observe how well the typical logistic function matches our classification objective by looking at the graph: if we optimize the values of \mathbf{w} and b suitably, the output of $f(\mathbf{x})$ may be interpreted as the likelihood of y_i being positive. For example, we would state that the class of \mathbf{x} is positive if it is more than or equal to the threshold of 0.5; otherwise, it is negative [10]. In practice, depending on the problem, the threshold may be chosen differently.

In logistic regression, the aim is to maximize the model's likelihood of correctly predicting the labels for our training set according to the model.

Maximum likelihood is the optimization criterion used in logistic regression [10]. According to our model, we now maximize the likelihood of the training data:

$$L_{\mathbf{w},b} \stackrel{\text{def}}{=} \prod_{i=1 \dots N} f_{\mathbf{w},b}(\mathbf{x}_i)^{y_i} (1 - f_{\mathbf{w},b}(\mathbf{x}_i))^{(1-y_i)}. \quad (10)$$

The formula $f_{\mathbf{w},b}(\mathbf{x})^{y_i} (1 - f_{\mathbf{w},b}(\mathbf{x}))^{(1-y_i)}$ may appear frightening, but it's only a clever way of stating " $f_{\mathbf{w},b}(\mathbf{x})$ when $y_i = 1$ and $(1 - f_{\mathbf{w},b}(\mathbf{x}))$ otherwise." If $y_i = 1$, then $(1 - f_{\mathbf{w},b}(\mathbf{x}))^{(1-y_i)}$ equals 1 since $1 - y_i = 0$ and we know that anything with a power of 0 equals 1. If $y_i = 0$, on the other hand, $f_{\mathbf{w},b}(\mathbf{x})^{y_i}$ equals 1 for the same reason [10]. We utilized the product operator \prod because the likelihood of observing N labels for N examples is the product of the likelihoods of each observation [10].

However, there is a problem with using this equation, for example, if we have three likelihood functions and we said earlier that the equation gives results from 0 to 1, for example, if we have half for all three likelihood functions, the function will multiply the three halves in each other and the result will be a too-small number and difficult to predict, so to avoid numerical overflow, we will use log-likelihood instead of likelihood. The log-likelihood is defined as follows:

$$\text{Log}L_{\mathbf{w},b} \stackrel{\text{def}}{=} \ln(L_{\mathbf{w},b}(\mathbf{x})) = \sum_{i=1}^N [y_i \ln f_{\mathbf{w},b}(\mathbf{x}) + (1 - y_i) \ln(1 - f_{\mathbf{w},b}(\mathbf{x}))]. \quad (11)$$

We used log here because the log is a **strictly increasing function**, maximizing it is the same as maximizing its argument, so the solution to this new optimization problem is the same as the original one [10].

2.4.1.2 Multi-Class Classification

Multiclass classification is the problem of classifying cases into more than two classes. [13]. In multi-class classification, examples are categorized into a set of specified categories, and one of the categories belongs to a specific example. There are two examples given above of multi-class classification, which are student grade and blood type example.

The most used algorithms in multi-class classification: k-Nearest Neighbors, Decision Trees, Naive Bayes, Random Forest and Gradient Boosting [12].

2.4.1.3 Multi-Label Classification

In certain cases, more than one label is suitable to describe a dataset example and this is where the term **multi-label classification** comes in [10]. There are several labels in multi-label classification that indicate the outcome of a specific prediction. An input may correspond to more than one label when creating predictions.

The difference between multiclass and multilabel classification is that multi-class classification assumes that each sample is assigned to only one label but in multi-label classification, more than one label is assigned to each sample.

Similarly, multiclass methods (decision trees, logistic regression, and neural networks, among others) may be applied to multi-label classification problems [10], because they provide the score for each class. [10].

2.4.2 Regression

Regression is a type of supervised learning that aims to predict a real-valued target given an unlabeled sample [10]. Predicting the temperature in a region, predicting a price or land area, and many more instances are all examples of regression.

A regression learning algorithm solves the problem by taking a set of labeled instances as inputs and producing a model that can accept an unlabeled example as input and output a target.

2.5 Ensemble Learning

A general machine learning approach in which multiple models' predictions are combined to achieve better performance [68]. Weak learners are the models that we use as input. They usually have a strong bias or too many variations and combining them gives us the best prediction model [69].

In Ensemble learning there are many ways that the goal can be achieved, the most 3 used classes in the practice of ensemble learning techniques are, Bagging, Stacking, Boosting.

2.5.0.1 Bagging

An ensemble technique that works by combining a group of weak learners designed beforehand to improve the classification accuracy. Each weak learner is different from another [76].

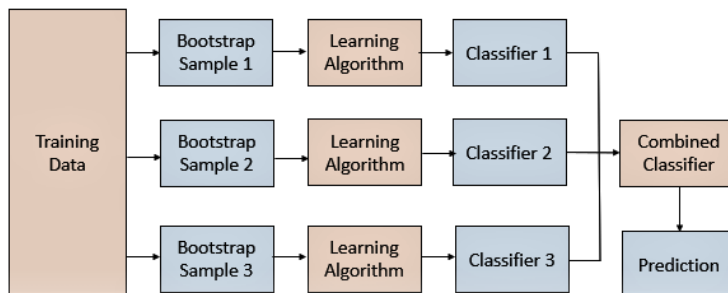


Figure 21: Bagging Ensemble [73].

In bagging, we work on models in parallel, where the weak learner's models are generated in parallel. When we merge the weak learners' models, the final model will be more accurate and have less variation [76]. Bagging also increases precision, and eliminates overfitting. It creates individual predictions for the ensemble by training each classifier on a random redistribution of the training set. The classifier is generated by randomly drawing with replacement, some of the samples that will be used again, while others will be omitted [77]. When an unknown instance is presented to each individual classifier, a majority or weighted vote is used to infer the class [78].

Random Forest is a bagging classification algorithm consisting of many decisions trees. You begin by dividing the training set into samples, then combining the test set with each training sample. Each combined training set and the test set will provide us with a prediction. Finally, we take the result and compare which is the most common result. The most common result is the final model. The random samples are denoted by B . We build the decision tree model f_b for each sample. The average of B forecasts is used to get the prediction for a fresh example x . [10].

$$y \leftarrow \hat{f}(x) \stackrel{\text{def}}{=} \frac{1}{B} \sum_{b=1}^B f_b(x). \quad (12)$$

2.5.0.2 Stacking

Stacking is a popular and a general method of using a high-level base learner to combine lower-level base learners to achieve greater predictive accuracy [80]. Stacking makes advantage of heterogeneous basis learner models, which means that multiple learning algorithms may be used to create the base learners.

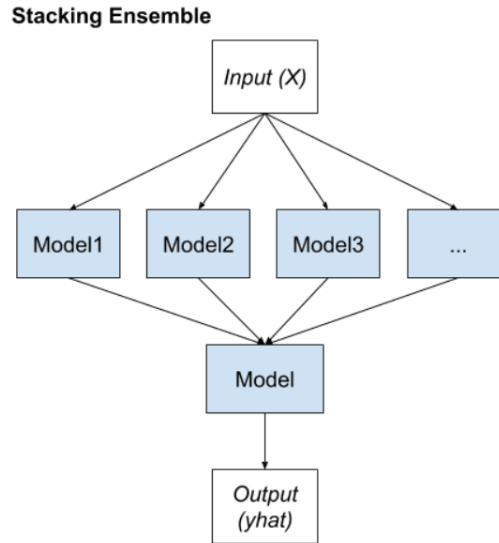


Figure 22: Stacking Ensemble [75].

Stacking can improve prediction accuracy while considering unbalanced datasets [81]. First, the algorithms are trained using the available data, then a combiner algorithm is trained to collect the predictions of the algorithms as input, original data will be divided into the trained dataset as shown in Figure 22, validation set, test set, we keep making predictions on the validation set using the training data. Also, use the module built on the training data to make predictions on the test data set. Because stacking is more difficult to analyze, it is used less frequently than bagging and boosting [80].

2.5.0.3 Boosting

An ensemble learning technique that learns from previous predictor mistakes to make better predictions in the future. Boosting starts with the original training data and iteratively creates multiple models by using a weak learner. The original training data will be combined with the new model, yielding a new model. Continuing to add models to the output of the previous one will yield the optimal model [10].

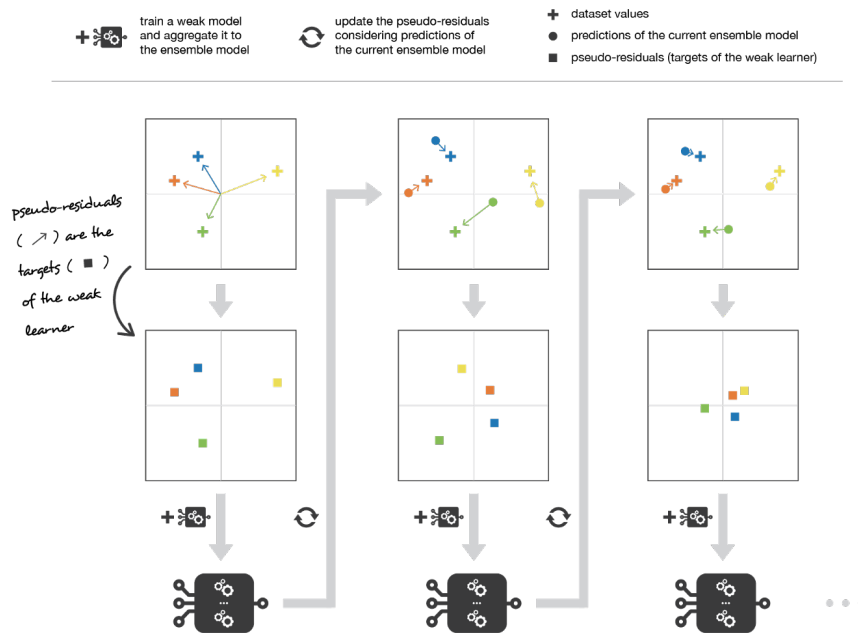


Figure 23: Boosting Ensemble [74].

The AdaBoost algorithm [79], which uses boosting, has been appointed as one of the top ten data mining algorithms. The algorithm is used to reduce bias. AdaBoost when training the classifier uses the whole dataset serially. Each model will wait until the previous one ends and take its output as input and focus on the difficult instances [78].

Gradient boosting is an ensemble learning algorithm, based on boosting, focused on building strong regressors.

$$f = f_0(x) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{i=1}^N y_i. \quad (13)$$

Start modifying labels of each example $i = 1, \dots, N$ in our training set as follows: $\hat{y} \leftarrow y_i - f(x_i)$ where \hat{y}_i , called the residual, is the new label, for example x_i . Now we use the modified training set, with residuals instead of original labels, to build a new decision tree model, f_1 . The boosting model is now defined as $f \stackrel{\text{def}}{=} f_0 + \alpha f_1$, where α is the learning rate [10].

Chapter 3: Literature Review

We will explore past literature relating to our problem in this chapter of our work, and we will present a summary of each scientific paper or article we read, noting that each summary discusses the objective, methodologies, and steps taken to produce the model and results. This chapter is divided into two subchapters: machine learning and deep learning-based solutions.

3.1 Machine Learning based solutions

The researcher's purpose in [32] is to investigate the prediction of Facebook users' personality traits using different features and measures from the Big 5 model. The XGBoost model was applied as the primary classifier as a baseline in this study. The study used three machine learners: Support Vector Machine (SVM), Logistic Regression, and Gradient Boosting. As a case study, the used dataset is the myPersonality dataset. The study was built with 250 users and 9917 status updates from the myPersonality sample. The dataset of Facebook users was labelled according to the Big 5 model. Each method was evaluated using single feature sets, such as SNA, LIWC, and SPLICE features. The XGBoost classifier outperforms the average baseline, with the highest prediction accuracy of 74.2%. Using the individual Social Network Analysis (SNA) features set, which achieved a higher personality prediction accuracy of 78.6%, the best prediction performance for the extraversion trait was achieved.

In [33], the main task was to predict the personality traits of a writer. Specifically, The task aimed at predicting authors' demographics based on its written tweets. The authors applied topic modeling using LDA [34] and J48 [35] decision tree as classification algorithm. In their experiments, they trained the classification models using 4 languages: English, Spanish, Italian and Dutch. The obtained results conducted on the the test dataset were encouraging with an accuracy of 69.7%. After 4 years in 2019, another Twitter-based paper focused on personality prediction, The researcher's goal in [36] was to predict personality based on the real-time set of tweets from Twitter using the Big Five model of personality traits. The researcher focused on Boosting algorithms because of their high accuracy and high performance. The advanced machine learning algorithms used in this paper are AdaBoost, LDA, and Multinomial Naïve Bayes. The researcher created the dataset using the Twitter streaming API. Twitter offers a Twitter streaming API for collecting real-time tweets from Twitter. The results showed that Multinomial Nave Bayes has the highest accuracy of 73.43, precision of 0.7, recall of 0.71, and F1-score of 0.72.

The researcher's goal in [37] is to classify a user's personality traits from input text using a supervised machine learning technique called the XGBoost classifier on the MBTI personality model. Dataset used is MBTI dataset. There are 8675 rows in this dataset, each representing a different user. Each user's past 50 social media posts, as well as their MBTI personality type, are included. Different machine learning classifiers, such as K-Nearest Neighbors (KNN), Decision Tree, Random Forest, Multi-Layer Perceptron (MLP), Logistic Regression (LR), Support Vector Machine (SVM), XGBoost, and Multinomial Naive Bayes (MNB), are used to determine personality traits after using class balancing algorithms on the imbalanced classes. The results show that all classifiers' scores across all personality traits are acceptable, the performance of the XGBoost classifier is exceptional. For I/E and S/N traits, they scored more than 99% precision and accuracy, while for T/F and J/P dimensions, they achieved about 95% accuracy. The KNN classifier, on the other hand, had lower overall performance.

In [38], the main task is to predict personality traits based on the user’s Facebook status updates using text semantic analysis and using the Big Five model of personality traits. The technique used in this study is the vector space model [40]. The relatedness score measures used in this work are Path-based measure [41] and information content-based measure [42]. The used dataset contains 9917 status posts for 250 Facebook users, released by the myPersonality project [43]. The results prove that the information content-based measure JCN [39] achieves the best average personality trait prediction with an accuracy of 64%. After that in 2021, another study [2] was made using the same dataset myPersonality project [13], The researcher’s goal in [28] was to try to predict a person’s personality by using their digital footprints in social media. The machine learning algorithms used in this paper are Convolutional Neural Network (CNN), Support Vector Machine (SVM), Naïve Bayes (NB), and Long short-term memory (LSTM). They used the myPersonality dataset to train the model, which contains around 10,000 statuses classified with Big 5 personality traits from 250 Facebook users. After trying all the algorithms, the Convolutional neural network (CNN) algorithm was the best with a score of 63.84%.

3.2 Deep Learning based solutions

The prediction system in this study [55] is built using some language features, as well as preprocessing steps and a variety of approaches. Multi-Layer Perceptron (MLP), Long Short Term Memory (LSTM) [54], Gated Recurrent Unit (GRU) [57], and 1-Dimensional Convolutional Neural Network (CNN 1D) [58] are the deep learning architectures used in this study [55]. In addition, we attempted to integrate the LSTM and CNN 1D architectures to create a new one [55]. A series of scenarios were conducted to produces the highest predication accuracy for each architecture. The highest accuracy when using the myPersonality dataset was 79.49% by using MLP architecture, while the highest accuracy when using the manually gathered dataset was 93.33% by using MLP and LSTM+CNN 1D architecture. A year later, another study was conducted, this time using one of the datasets from the previous mentioned paper, the myPersonality project datasets. This study [29] proposes utilizing online posts to predict personality using deep learning. In this paper [29], a new structure called attRCNN [29] is designed by introducing an attention mechanism and a batch normalization technique to the typical conventional neural network to perform the vectorization of a single text post and combine the attRCNN structure with a variant of CNN-based Inception structure through a hierarchical architecture to learn deep semantic representations of the aggregation of each user’s text posts. The final feature space is constructed by concatenating the learnt deep semantic representations with pre-extracted statistical linguistic features vectors and adapting it to the Gradient Boosting Regression(GBR) algorithm [30] to predict the big five personality scores. The dataset was collected as part of the myPersonality project [31]. The experimental evaluation showed that taking deep semantic representations as input to regression algorithms contributes a lot to better predictions.

In this paper [44], a pre-trained multi-model trinity architecture with model averaging was proposed, namely a trinity of classifiers BERT [45-46], RoBERTa [47], and XLNet [48]. The personality framework used is The Big Five. Two datasets were used: an expanded dataset from previous research [47] with a manually collected Indonesian Twitter dataset added, which totals 502 users and 46,238 statuses, and the open-source myPersonality Facebook dataset [31, 49], which includes 250 users and 9917 statuses. The data was split into 3 separate sets: training, testing, and validation sets (70%, 15%, 15%). The two datasets were preprocessed by removing URLs, symbols, stop-words, clitics, emoticons, and affixes using stemming. The contractions were also expanded and all words were normalized to be lowercase. But in the case of the Twitter dataset, an extra step of translating Bahasa to English was done. Additionally, two feature extraction methods were used: pre-trained model features and statistical features. The former method includes the pre-trained models BERT, RoBERTa, and XLNet. The latter method uses TF-IGM [50] as a term weighting factor instead of TF-IDF, and employs sentiment analysis and the NRC lexicon database [51]. Figure 24 shows the architecture in detail. The model evaluation metrics used are accuracy and F1-score, for which they have scored 77.34% and 0.749 respectively for the Facebook dataset, and 77.34% and 0.760 respectively for the Twitter dataset, all Facebook dataset.

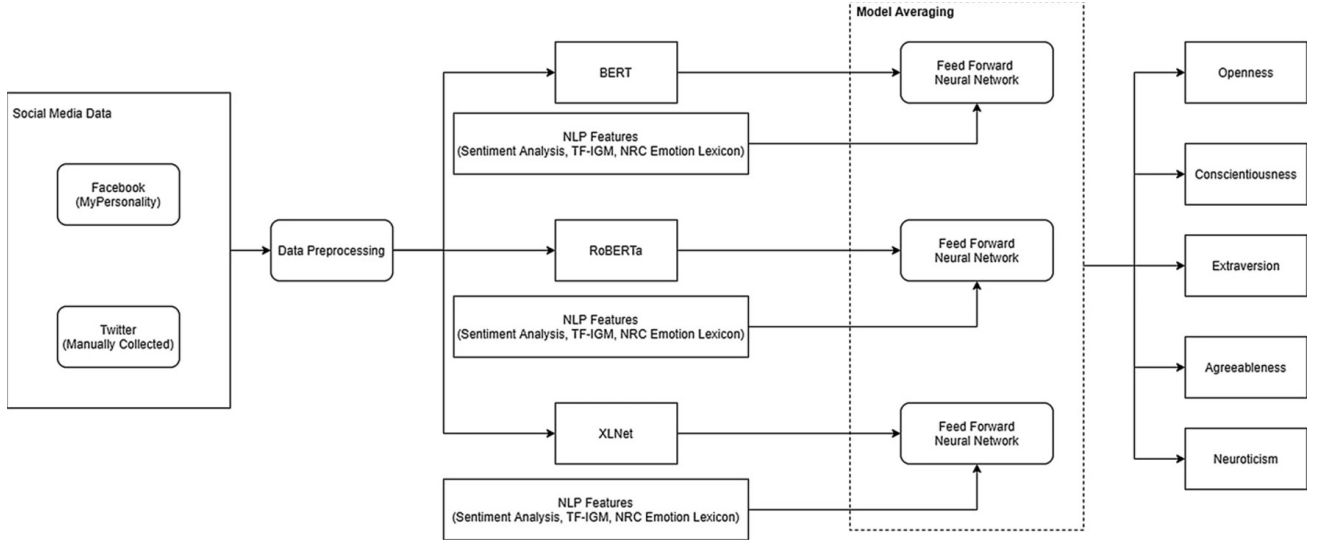


Figure 24: The architecture proposed in [24].

This research [35] aims at building an LSTM-based prediction model that can predict personality traits from a person’s social media footprint. The dataset used in this study was gathered from Twitter and contains 11,769,202 tweets written by 5081 users [35]. The tweets of 5081 users were converted to the IBM Personality Insights service’s input format, resulting in the collection of 35 personality traits for each user, which were grouped under the five traits of the Big Five model [52] according to their relevance to the respected trait. In addition to the previously described dataset, the PAN-2015-EN Personality Dataset is utilized as a benchmark dataset [35]. The dataset is separated into two datasets, training and testing, containing a total of 27,343 tweets. In this paper, the basic LSTM model [54] is used with a variation of the LSTM model called Bidirectional LSTM(Bi-LSTM) [53].

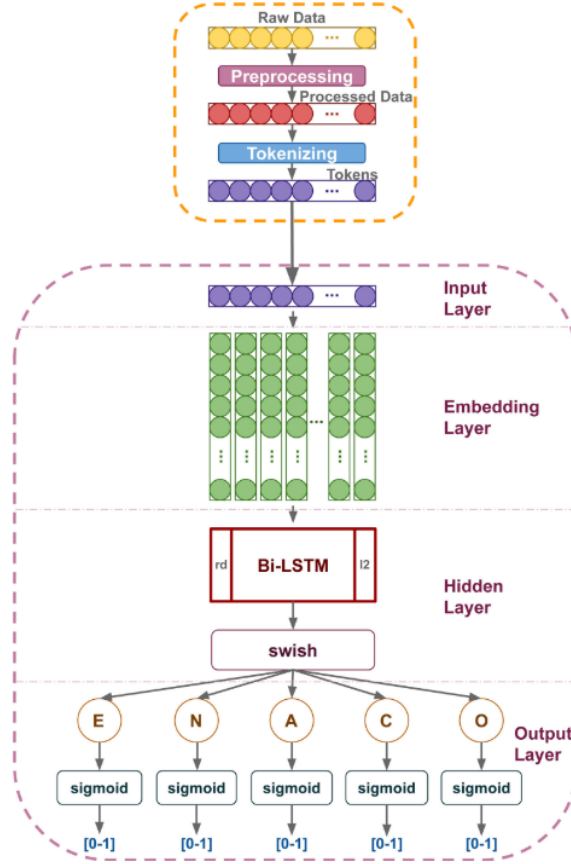


Figure 25: A structural representation of the model (rd: recurrent dropout, l2: l2 regularizer) [35].

To begin, the tweets were tokenized and a dictionary was generated; then, each tweet in the dictionary was vectorized, and the vectorized tweets were fed to the model as input. To analyze the results and find the best one, an analysis was carried out using various combinations of the model, preprocessing procedures, and hyperparameter tuning operations. The Bidirectional LSTM gives superior outcomes than the basic LSTM as a result of the analysis based on the LSTM type.

Table 1: Summary of Literature Review

Ref #	Goal	approach	Dataset	Result
[29]	Build a prediction system that leverages social media posts to forecast the author’s personality using a deep learning based approach.	Gradient Boosting Regression, Support Vector Regression, Random Forest and Multi-Layer Perceptron	myPersonality project	SVR, GBR, RF, and MLP average MAEs are 0.42796, 0.42768, 0.43060, and 0.53162, respectively.
[28]	Predicting the personality by using the digital footprints of social media users.	Convolutional neural network (CNN), Vector Machine (SVM), Naïve Bayes (NB), Long short-term memory (LSTM).	myPersonality project	CNN = 63.84%.
[37]	Build a prediction system that classifies personality traits based on text using a machine learning approach.	KNN, Decision Tree, Random Forest, MLP, Logistic Regression (LR), Support Vector Machine (SVM), XGBoost, and MNB.	MBTI Dataset	XGBoost I/E = 99% S/N = 99% T/F = 95% J/P = 95%
[32]	build a prediction system to predict the Facebook users’ personality traits using different features and measures from the Big 5 model.	(SVM), Logistic Regression, and Gradient Boosting. as a feature sets SNA, LIWC, and SPLICE features.	myPersonality dataset	average = 74.2%. individual SNA features set = 78.6%
[33]	Predicting the personality traits of a writer from his written tweets.	topic modeling, J48.	PAN 15’ dataset	Predicting the personality with an accuracy of 69.7%

Table 2: Summary of Literature Review

Ref #	Goal	approach	Dataset	Result
[36]	Build a prediction system based on the real-time set of tweets	AdaBoost, LDA, Multinomial Naïve Byes, Boosting.	Twitter streaming API.	Multinomial Nave Bayes has the highest accuracy of 73.43, precision of 0.7, recall of 0.71, and F1-score of 0.72.
[38]	Build a personality prediction system from Facebook status updates	Vector Space, semantic similarity metrics	myPersonality project	information content-based measure JCN achieved the best accuracy value of 64%.
[35]	Build a prediction model that uses social media data to predict personality traits	The basic LSTM architecture with a variation of LSTM called Bi-LSTM	PAN-2015-EN Personality Dataset Twitter dataset	Twitter dataset LSTM = 0.1920 Bi-LSTM = 0.1879 PAN-2015-EN dataset Bi-LSTM = 0.1688. LSTM = 0.1693
[44]	build personality prediction system based on text using user statuses.	BERT [17-18], RoBERTA [19], and XLNET [20], TF-IGM, sentiment analysis, and NRC	myPersonality dataset manually collected Indonesian Twitter dataset.	Facebook dataset accuracy = 77,34% F1-score = 0.749 Twitter dataset accuracy = 77.34% F1-score = 0.760
[55]	Build a personality prediction system that can predict based on Facebook activities using deep learning architectures.	MLP, LSTM, GRU, CNN 1D and a combination of LSTM and CNN 1D.	myPersonality project manually gathered dataset.	myPersonality dataset MLP = 79.49% manually gathered data MLP and LSTM+CNN = 93.33%.

Chapter 4: Proposed Work

Our project’s goal is text-based personality prediction using the Big Five personality traits taxonomy. To this aim, we’re proposing a deep learning architecture shown in Figure 26 that inputs preprocessed myPersonality data into three state-of-the-art transformers: XLNet, BigBird, and ConvBERT. Additionally, sentiment analysis is utilized alongside the NRC emotion lexicon. The resulting data is then fed to feed-forward neural networks and then averaged out for the final prediction. The method of averaging is proposed to maximize accuracy by tending to whatever faults that come about from the other two transformers, such as in the case of BigBird being optimal for long texts and relatively non-optimal for shorter texts. Here, XLNet and ConvBERT, which are better suited for shorter texts, can tend to the faults of BigBird for example, or vice versa for longer texts, through averaging.

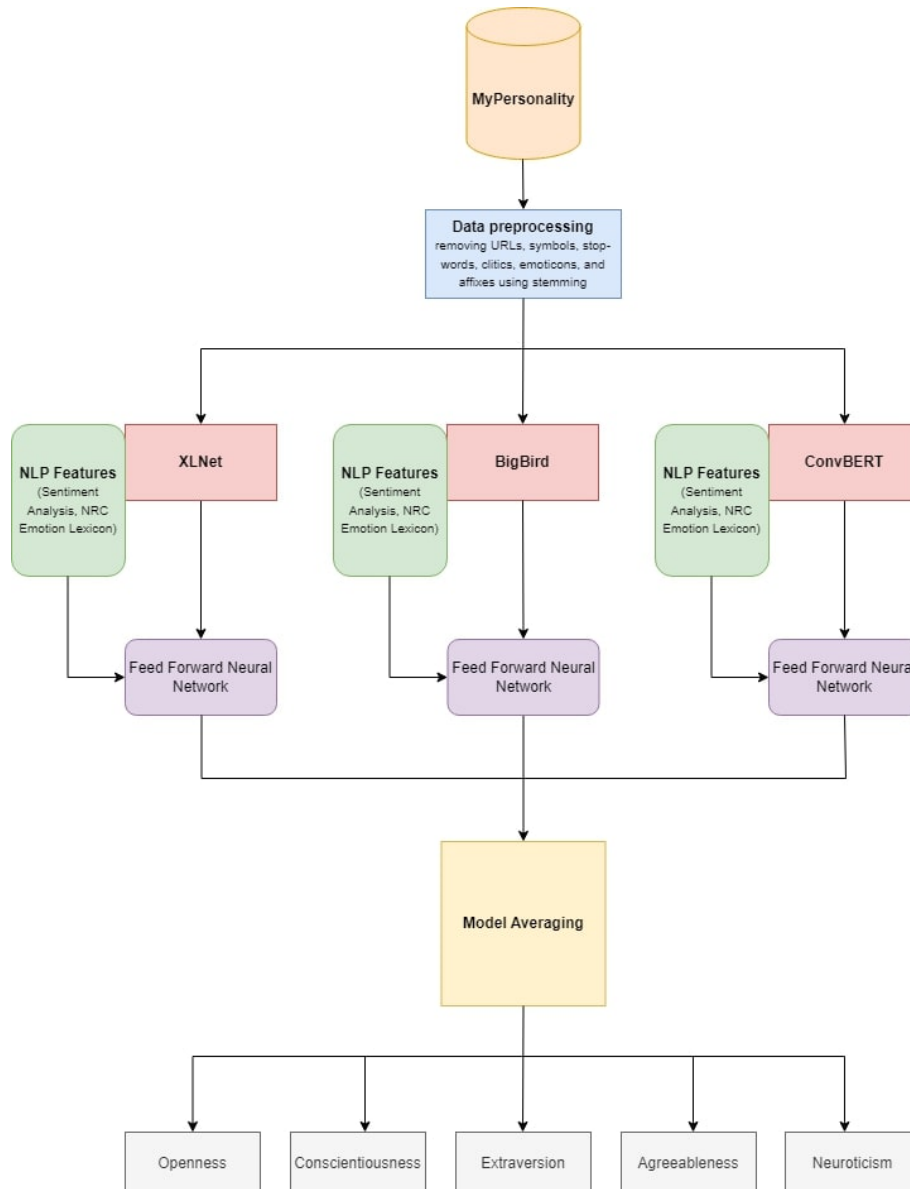


Figure 26: The architecture of the proposed system inspired by [44].

4.1 Preprocessing

Preprocessing is the preparation of raw data in order to make it suitable for building and training models. We plan to preprocess the data similarly to the research in [117] where collected

data is cleaned starting by removing the URLs, symbols, and emoticons in the social media status, then the contractions will be expanded, each sentence will be normalized to be of lower-case form, and ambiguities will be resolved by removing stop-words and clitics. Finally, a stemming or lemmatization function will be used to remove all affixes and to have a unified representation.

4.2 Sentiment Analysis

Sentiment analysis is the method of determining an author’s feelings and emotions by splitting the author’s text into positive, negative, and neutral components. These components can help us determine specific feelings and emotions such as anger, happiness, sadness, etc. Sentiment analysis has become one of the most active research areas in natural language processing (NLP) [118].

4.3 NRC Emotion Lexicon

The NRC Emotion Lexicon is a lexicon that maps English words to eight basic emotions, namely anger, fear, anticipation, trust, surprise, sadness, joy, and disgust, and two sentiments negative and positive. The lexicon was constructed using manual crowdsourcing [153-154].

4.4 XLNet

XLNet is a generalized autoregressive pretraining method, also called a transformer, that builds upon Bidirectional Encoder Representations from Transformers (BERT) [48, 45] and Transformer-XL [48, 121]. It learns bidirectional contexts by taking permutations of the factorization order and maximizing the expected log likelihood over them [48]. And, thanks to its autoregressive formulation, it bypasses the limitations present in BERT. It integrates some ideas from Transformer-XL such as its segment recurrence mechanism and relative encoding scheme into pretraining, improving the performance especially on tasks that involve long sequences of text. XLNet consistently outperforms BERT on many tasks, such as those that involve The General Language Understanding Evaluation (GLUE) [48, 122] dataset, reading comprehension on SQuAD [48, 123] and RACE [48, 124], text classification on Yelp [48, 125] and IMDB [48, 126], document ranking on ClueWeb09-B [48, 127], and others [48].

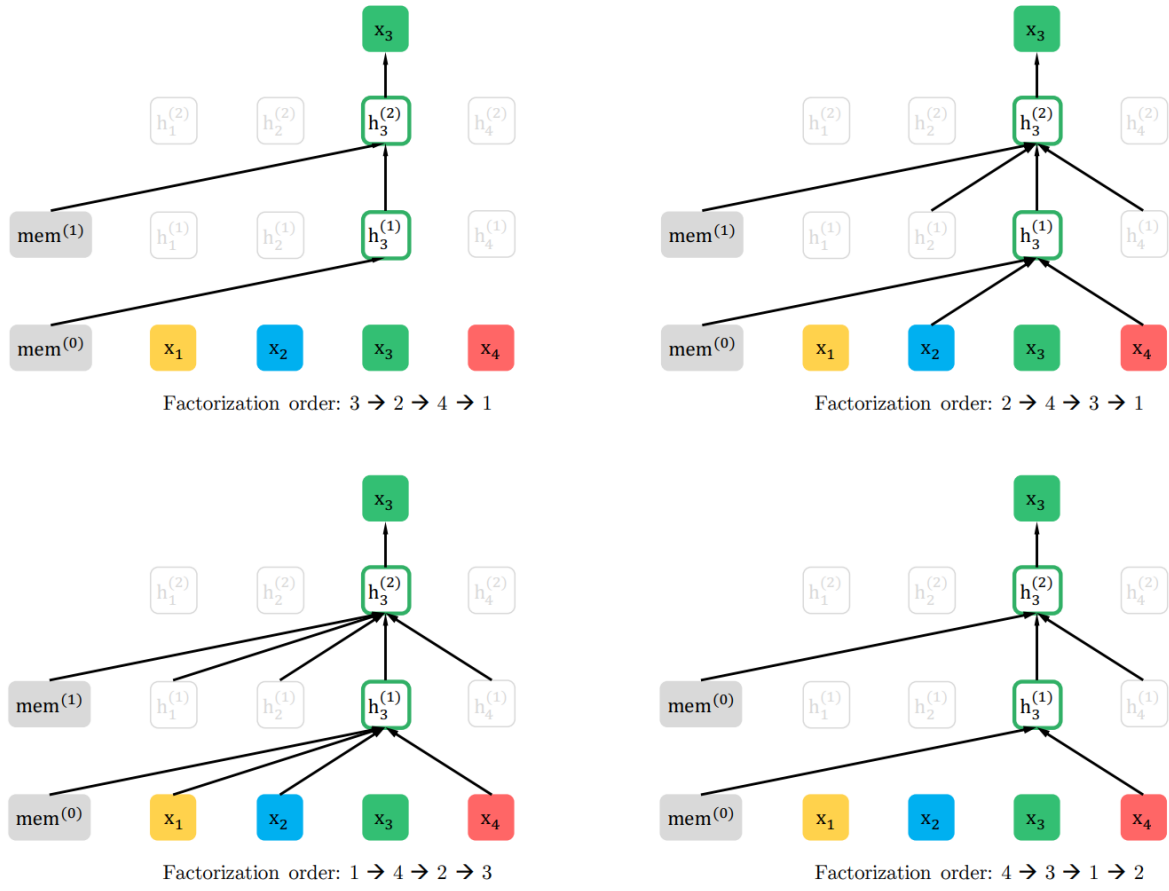


Figure 27: Permutation language modeling with predicting x_3 as the task given the same input sequence x but with different factorization orders. The order is randomly permuted and each token’s prediction is based on its preceding tokens [48].

4.5 BigBird

BigBird [128] is a transformer that uses a sparse attention mechanism as opposed to a full one so as to bypass the quadratic dependency (mainly in terms of memory) on sequence length present in other transformers. Its architecture is the same as BERT and was initialized with RoBERTa, only differing in the attention mechanism. Its sparse attention can handle sequences 8x longer than what was previously possible with similar hardware and still performs better, especially in the tasks of question answering and document summarization. As illustrated in Figure 28, its novel sparse attention mechanism consists of three main parts [128]:

- A set of g global tokens attending on all parts of the sequence.
- All tokens attending to a set of w local neighboring tokens.
- All tokens attending to a set of r random tokens.

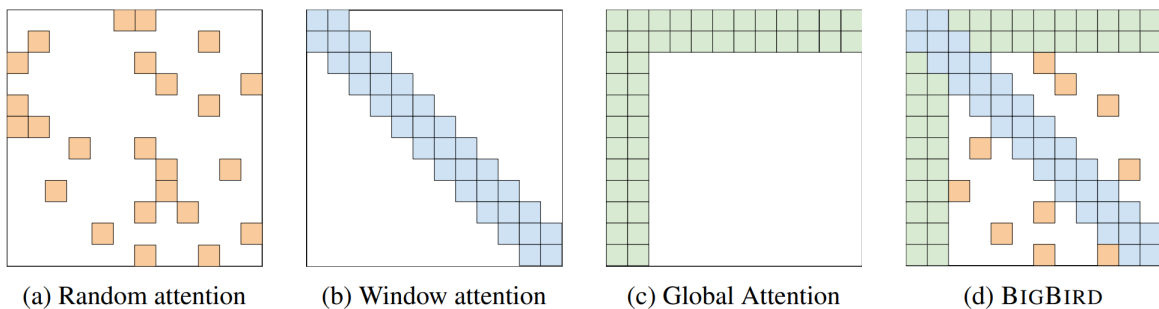


Figure 28: (a) (b) (c) building blocks, (d) culminating in BigBird’s attention mechanism. White color denotes the absence of attention. (a) $r = 2$, (b) $w = 3$ (c) $g = 2$. (d) the combined BigBird model [128].

4.6 ConvBERT

ConvBERT [129] is another variant of BERT that directly deals with a number of its problems, such as the computation redundancy as a result of its attention heads querying on the whole input sequence when some attention heads only need to learn local dependencies. It has a mixed attention module in place of the self-attention heads present in BERT, given convolution’s capability of better capturing local dependencies. It also dynamically generates the convolution kernel by using a novel span-based dynamic convolution operation on multiple input tokens. The aforementioned span-based dynamic convolution and mixed attention bring about the model ConvBERT. With a score of 86.4 on the GLUE benchmark, ConvBERT outperforms BERT (score of 80.9) and ELECTRA (score of 85.7) [130], in their base forms, and does so using less than 1/4 of the latter’s training cost.

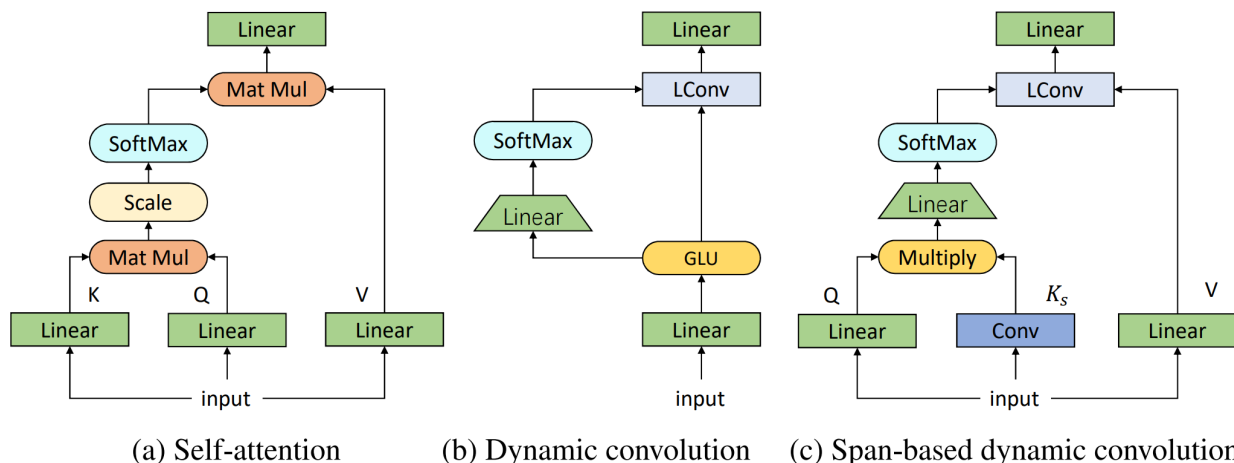


Figure 29: Illustration of self-attention, dynamic convolution and span-based dynamic convolution [129].

Figure 29.a illustrates self-attention which is BERT’s basic building block. It models global dependencies in the input sequence of tokens. Figure 29.b illustrates dynamic convolution which can better generate kernels conditioned on the input token. It’s suitable for modelling local dependency with its only disadvantage being the kernel’s dependency on a single token. Figure 29.c illustrates the span-based dynamic convolution which generates the local relation of the input token conditioned on its local context instead of a single token, and to make the convolution compatible with self-attention, a linear transformation is applied to generate query Q and value V, and depth-wise separable convolution generates “span-aware” Ks. The result of point-wise multiplication of query Q and span-aware key Ks pairs is then used to generate the dynamic convolution kernel. LConv denotes a light-weight depth-wise convolution that ties all weights along channel dimension, using a different convolution kernel at each position.

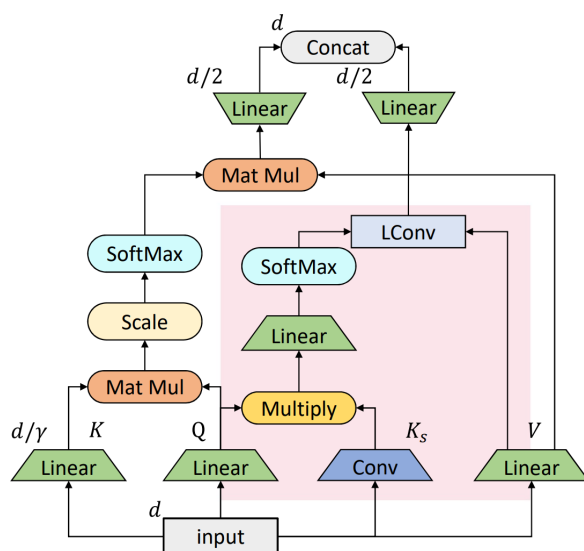


Figure 30: Illustration of mixed-attention block [129].

In Figure 30, highlighted in pink is the mixture of self-attention and span-based dynamic convolution. The Query is shared but Keys are different for each so as to generate the attention map and convolution kernel respectively. d is the embedding size of the input and r is the reduction ratio.

4.7 Feed-Forward Neural Network

The feed-forward neural network uses the pre-trained model features and statistical features as input. Each neural network is comprised of three connected layers with alternating Rectified Linear Unit (ReLU) activation functions and batch normalization. In the averaging model function, the output of feed-forward neural networks will be added.

Chapter 5: Experimental Settings

5.1 Dataset

In 2007, David Stillwell launched a Facebook application called MyPersonality project [31]. The idea of the application is to give the user real psychometric tests and get the results instantly. Around 40% of the respondents agreed to share data from their Facebook profiles, leading him to create one of the world's largest social science research databases. MyPersonality is an open-source dataset consisting of 250 users with a total of 9917 statuses.

5.2 Performance Measure

5.2.1 Precision

The precision is the proportion of correctly predicted positive outcomes to the total number of positive outcomes [10].

$$precision \stackrel{\text{def}}{=} \frac{TP}{TP + FP}. \quad (14)$$

5.2.2 Recall

The recall is the proportion of correct positive predictions to the total number of positive cases in the dataset [10].

$$recall \stackrel{\text{def}}{=} \frac{TP}{TP + FN}. \quad (15)$$

5.2.3 Accuracy

The accuracy is calculated by dividing the number of correctly classified examples by the total number of classified examples. It is provided by in terms of the confusion matrix [10].

$$accuracy \stackrel{\text{def}}{=} \frac{TP + TN}{TP + TN + FP + FN}. \quad (16)$$

5.2.4 F1 Score

By considering their harmonic mean, a classifier's precision and recall are combined into a single metric. Its primary application is to compare the performance of two classifiers [10].

$$F1score = 2 * \frac{recall * precision}{recall + precision}. \quad (17)$$

Table 3: Multiclass example to calculate the four concepts above.

		True/Actual		
		Apple	Orange	Banana
Predicted	Apple	3	4	2
	Orange	1	3	1
	Banana	0	3	6

In the multi-class case, if we want to analyze a class, the class will be positive and the others negative.

For Apple:

- 1- **Precision:** By using the Precision equation, we get the following result: $\frac{3}{3+1+0} = 75\%$
- 2- **Recall:** By using the Recall equation, we get the following result: $\frac{3}{3+4+2} = 33\%$
- 3- **Accuracy:** By using the Accuracy equation, we get the following result: $\frac{3+3+1+3+6}{3+4+2+1+3+1+0+3+6} = 70\%$
- 4- **F1 score:** By using the F1 score equation, we get the following result: $\frac{2*0.33*0.75}{0.33+0.75} = 46\%$

For Orange:

- 1- **Precision:** By using the Precision equation, we get the following result: $\frac{3}{3+4+3} = 30\%$
- 2- **Recall:** By using the Recall equation, we get the following result: $\frac{3}{3+1+1} = 60\%$
- 3- **Accuracy:** By using the Accuracy equation, we get the following result: $\frac{3+3+2+6+6}{3+4+2+1+3+1+0+3+6} = 61\%$
- 4- **F1 score:** By using the F1 score equation, we get the following result: $\frac{2*0.6*0.3}{0.6+0.3} = 40\%$

For Banana:

- 1- **Precision:** By using the Precision equation, we get the following result: $\frac{6}{6+1+2} = 66\%$
- 2- **Recall:** By using the Recall equation, we get the following result: $\frac{6}{6+3+0} = 66\%$
- 3- **Accuracy:** By using the Accuracy equation, we get the following result: $\frac{6+3+4+1+3}{3+4+2+1+3+1+0+3+6} = 74\%$
- 4- **F1 score:** By using the F1 score equation, we get the following result: $\frac{2*0.66*0.66}{0.66+0.66} = 66\%$

Table 4: The results of the classes by the four evaluation models:

Class	Precision	Recall	Accuracy	F1 score
Apple	75%	33%	70%	46%
Orange	30%	60%	61%	40%
Banana	66%	66%	74%	66%

5.3 K-Fold Cross Validation

Cross-validation is a procedure used to evaluate machine learning models on limited data. Dividing all the data into K groups of data, for each unique group, take the group as a test data set. Then take the remaining groups as a training data set. After that, fit a model on the training set and evaluate it on the test set retain the evaluation score and discard the model. Finally, summarize the results. In the end, each data block is used for testing and training.

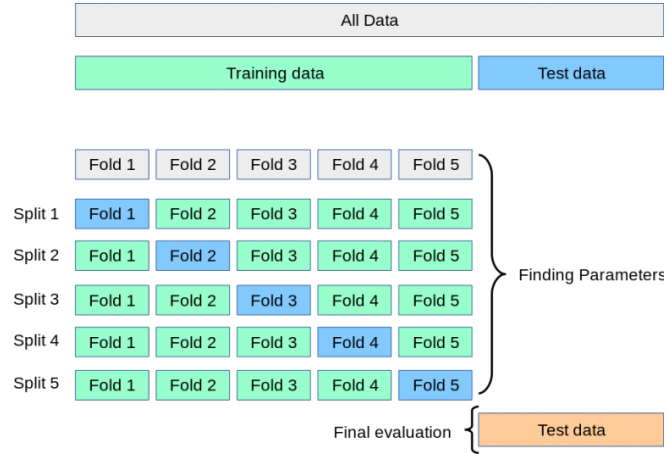


Figure 31: K-fold cross-validation with $K = 5$ [118].

Chapter 6: Experiments

In this chapter we will document the results of the various techniques we have mentioned up to this point, and we will try to explain why certain results occur, building up all the way to the final proposed model. We will also explain the difficulties faced in the implementation environments and in the model-building process.

6.1 Experiments' Environment

The proposed system was implemented and tested using Python 3.11 through various Integrated Development Environments (IDEs) and environments including but not limited to Visual Studio Code [142-143], JupyterLab and Jupyter Notebook [144-146], but Google's Colaboratory bore the brunt of the training pipelines [147]. In tasks besides training, the code was run and tested using a computer with 16GB OF RAM and a 3.6 GHz, 8 core, and 16 threads processor.

The libraries utilized are PyTorch, NumPy, pandas, Huggingface's Transformers, Huggingface's Datasets, sklearn, and Fast-ML [135-141].

6.2 Experiment Issues

As a consequence of the enormity of Transformer models, we had to resort to Google Colab to utilize its cloud computing. The most memory-intensive of these models is Big Bird as it cannot be trained by standard Colab GPUs. Even with capable hardware, the feedback loop for deep learning tasks is especially slow relative to shallow learning tasks, making hyperparameter tuning and other adjustments costly in terms of time and computation. We used a different base for BigBird with 79M parameters instead appropriately named DistilBigBird [134] but fine-tuned on Persian language tasks, however, due to the relatively language-agnostic configuration of Transformers, we can fine-tune this task on English texts.

Other complications arose in the attempt to implement custom behavior in the Application Programming Interfaces (APIs) of various libraries (Huggingface's Transformers, sklearn, PyTorch etc.) due to the documentation lacking in some aspects and the implementation inflexible in others. For these instances, we manually implemented behavior on top or in correspondence with the already present APIs.

6.3 Data Exploration

It's important to point out that the mypersonality dataset takes trait indicators as continuous numbers (4.35/5, 1.5/5, etc.) and the categorical features are merely one-hot encoded representations of the latter. The thresholds at which a trait is considered existent in a person are variable across all the traits. To be exact, to have any of the following traits as per the mypersonality dataset, one needs to have scores satisfying the constraints in Table 5.

Table 5: Trait Classification Criteria

Trait	Score >
Open	3.75/5
Agreeable	3.55/5
Conscientious	3.50/5
Extraverted	3.45/5
Neurotic	2.75/5

Furthermore, as apparent in Figure 32, there is a considerable class imbalance that will surely undercut the performance of the models, especially in the case of Openness and Neuroticism, where there is a 74%-26% and 37%-63% class distribution respectively. To address this imbalance, we will try undersampling later to alleviate the losses.

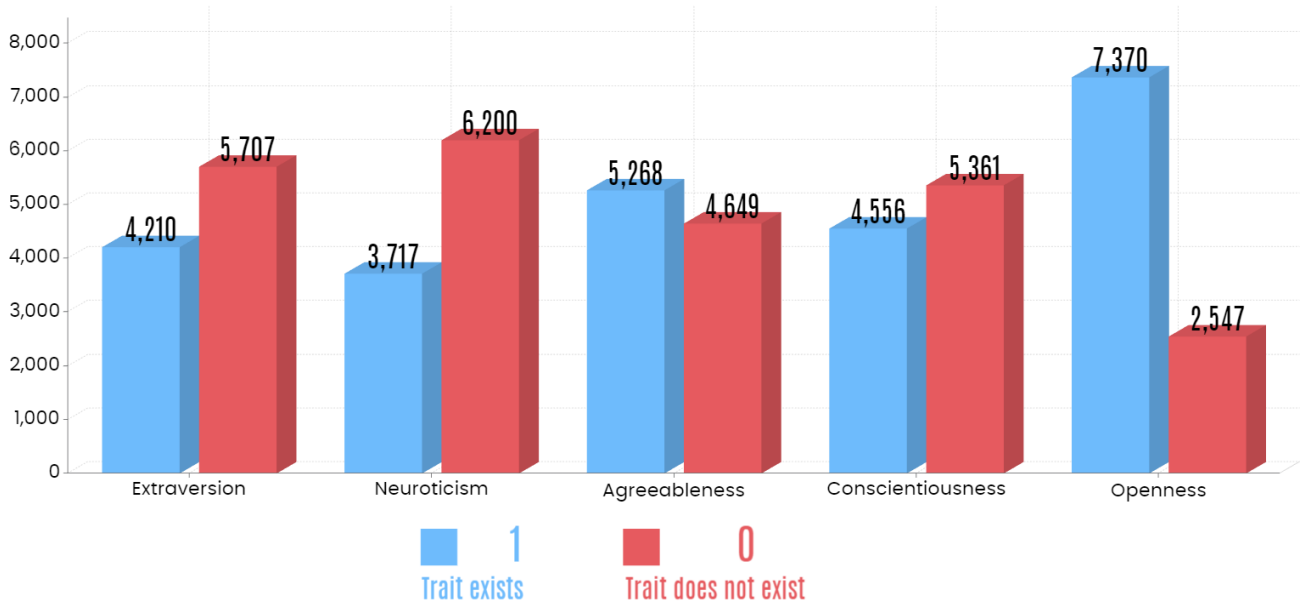


Figure 32: Class imbalance in the dataset.

Another consideration regarding imbalances is the over-representation of a few authors in the number of statuses in the dataset. Although the number of authors is 250, approximately 10% of statuses were generated by a measly 5 users, 25% by 15, and 50% by 41. This makes for a right-skewed distribution as shown in Figure 33.

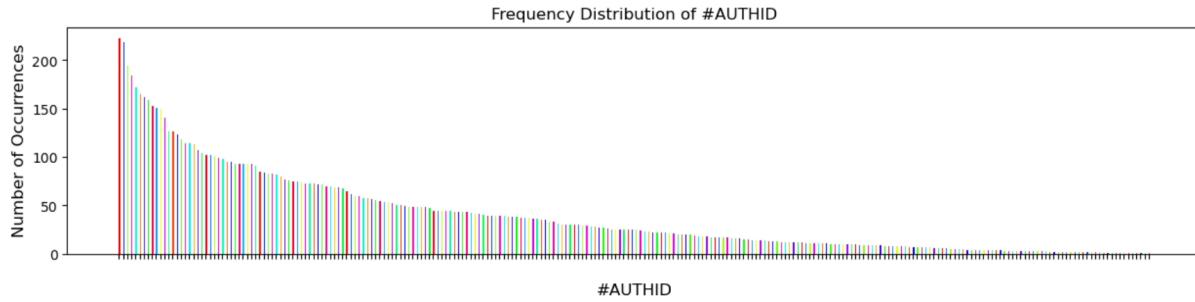


Figure 33: The relation between authors and the text in the dataset makes a right-skewed distribution.

Chapter 7: Results

The models we are using are DistilBigBird, ConvBERT, XLNet, and in cases where we mention model averaging we’re essentially averaging the normalized logit outputs of the three models and then applying a sigmoid activation function on the results. The metrics we used to evaluate the training of the models include F1 with micro, macro, weighted, and sample averaging; and accuracy, precision, recall, and specificity. If no specific averaging is mentioned, assume it is of micro form.

7.1 Model Hyperparameters

7.1.1 Maximum Sequence Length

Sequence length is the length of the input data sequence. With Transformer models, the sequences we can pass to the model have a maximum length. Most models can handle up to 512 or 1024 token sequences, and it will not work when processing sequences longer than the maximum. When the maximum length is increased, it will help the model process a larger text, but it will reduce the speed of the model, and vice-versa. Our maximum length is the default for each model’s tokenizer, but it is dynamically allocated across batches through dynamic padding. The data will be split into batches, and each datum in a batch will have to be of the length of the batch it’s in, so different batches will have different sequence lengths, but no batch will have a sequence length higher than the maximum.

7.1.2 Number of Epochs

An epoch is an entire pass through the neural network (Forward propagation) and back (Backpropagation) using the entire training dataset [151]. Upon training the XLNet model, we can see—as shown in Figure 34—that the validation loss and training loss start at relatively the same quantity and slowly decrease until they bifurcate in opposite directions with validation loss increasing upwards, which indicates that the model starts to overfit from there onwards. Showing an early “U” shape usually indicates the learning rate is high. We see that the divergence between the training and validation loss starts to happen at epoch 4, so epoch 4 can be used as a viable stopping point for XLNet with a learning rate of $3.00e-5$ and a linear learning rate scheduler. However, we apply a sigmoid activation on the normalized logits with a threshold of 0.5 to decide the class, so we will rely on the metrics we really care about (F1, Accuracy, etc.) because they are more resistant to overfitting than validation loss so we can carry on training beyond that point especially as long as these metrics are good on the test set. Solutions to mitigate overfitting abound, so we’ll attempt a few. One solution is to decrease the learning rate.

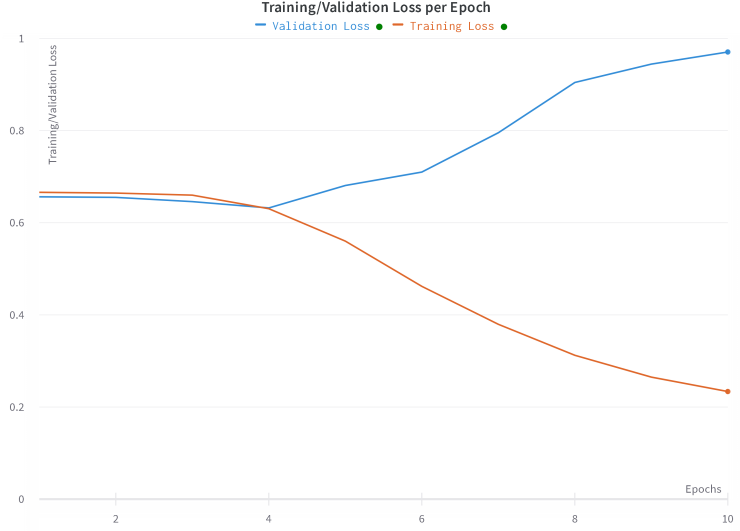


Figure 34: Training and validation loss per epoch.

We can see the difference in the following plot. After decreasing the learning rate from “3.00e-05” to “2.01e-05”. We see that the divergence between the training and validation loss starts to happen at epoch 6. We can control the overfitting problem by decreasing the learning rate over more epochs.

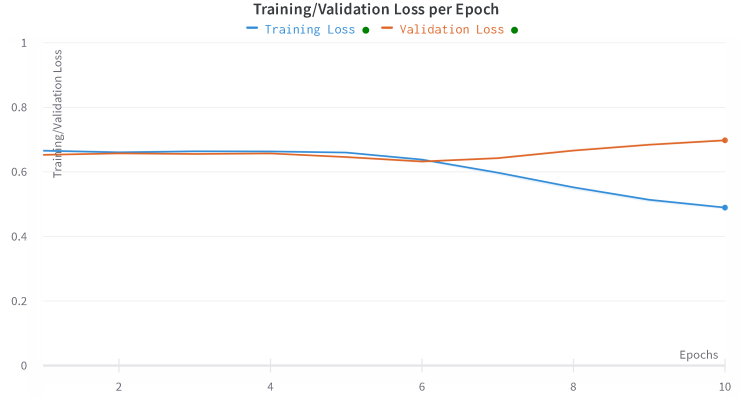


Figure 35: Training and validation loss per epoch.

7.1.3 Batch Size

Batch size is the size of the training examples to be propagated through the neural network in one iteration [149]. Generally, using higher batch sizes increases memory consumption. For this very reason, we were only able to experiment with 4 different batch sizes due to the scarcity of computational resources and time. As shown in Table 6, the difference between batch sizes is relatively insignificant, but a batch size of 8 has the best results across all metrics. We will therefore use a batch size of 8 from this point onwards.

Table 6: Results of different batch sizes using model averaging.

Batch size	F1 Micro	Average Accuracy	F1 Macro	F1 Weighted	F1 Samples
4	66%	65%	60%	64%	62%
8	68%	67%	63%	66%	64%
16	67%	66%	62%	65%	63%
32	67%	66%	61%	65%	63%

7.1.4 Learning Rate

The learning rate hyperparameter is responsible for adjusting the network's weights in relation to the loss gradient [150]. The higher the learning rate the more likely learning will overstep minima, and conversely, the lower the learning rate the more likely for learning to get stuck in local minima and converge slower [152].

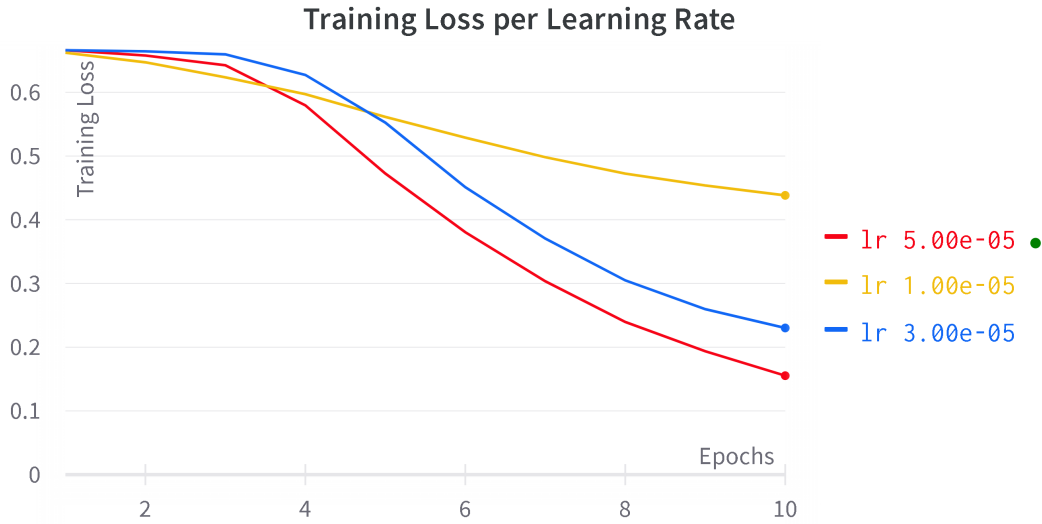


Figure 36: Training loss per learning rate.

A Learning rate value of $3.00e-05$ has the advantage in Figure 36, and a higher learning rate value of $5.00e-05$ will cause early overfitting, whereas a value of $1.00e-05$ will make the model converge very slow.

Table 7: Results of Learning rates using a batch size of 8 and using model averaging.

Learning rate	F1 Micro	Average Accuracy	F1 Macro	F1 Weighted	F1 Samples
1.00e-05	64%	64%	59%	63%	61%
3.00e-05	68%	67%	63%	66%	64%
5.00e-05	62%	62%	51%	56%	57%

Looking at Table 7, it seems that up until a certain point, raising the learning rate boosts the performance model, and then it takes a massive drop. Given that a learning rate of $3.00e-05$ yields the highest results, we will use it from this point onwards.

7.1.5 Learning Rate Scheduler

A learning rate scheduler is any function that dynamically alters the learning rate during training per a pre-defined schedule [134]. Our standard training routines had the hyperparameter set as "linear". Table 8 shows the results of applying a linear learning rate scheduler, in contrast to Table 9 with a "cosine" scheduler.

Table 8: Results of a linear rate scheduler using $3.00\text{e-}05$ as the learning rate and batch size of 8.

Models	F1 Micro	Average Accuracy	F1 Macro	F1 Weighted	F1 Samples
DistilBigBird	64%	62%	59%	63%	60%
ConvBERT	66%	65%	62%	65%	62%
XLNet	66%	64%	62%	65%	62%
Averaged Model	68%	67%	63%	66%	64%

Table 9: Results of changing the learning rate scheduler to be of cosine form using $3.00\text{e-}05$ as the learning rate and a batch size of 8.

Models	F1 Micro	Average Accuracy	F1 Macro	F1 Weighted	F1 Samples
DistilBigBird	61%	60%	56%	59%	56%
ConvBERT	66%	65%	62%	65%	62%
XLNet	66%	64%	62%	66%	63%
Averaged Model	68%	67%	63%	66%	64%

As evident in the contrasting results of Table 8 and Table 9, DistilBigBird took a relatively noticeable hit across all metrics when a cosine learning rate scheduler was applied to its training, whereas ConvBERT slightly degraded in all metrics but average accuracy where it slightly increased, and XLNet improved slightly across all metrics. We will therefore only alter the learning rate scheduler in the case of XLNet to be of cosine form from this point onwards.

7.1.6 Weight Decay

Weight decay is a popular deep learning regularization technique applied to neural network weights. Specifically, it is used to regularize the sizes of the weights of specific parameters, which may enhance prediction performance and combat overfitting [148]. Thus far we have only trained with weight decay set as 0. As one of the ways to combat overfitting, we saw it fit to try out adding a regularization term in the form of weight decay with a value of 0.001.

Table 10: Results of having a weight decay value of 0.001, with a learning rate value of $3\text{e-}5$, a cosine learning rate scheduler, and a batch size of 8.

Models	F1 Micro	Average Accuracy	F1 Macro	F1 Weighted	F1 Samples
DistilBigBird	63%	62%	59%	62%	60%
ConvBERT	66%	65%	61%	65%	62%
XLNet	63%	62%	57%	61%	59%
Averaged Model	67%	66%	61%	65%	63%

As apparent in Table 10 in contrast to Table 9, DistilBigBird with a weight decay of 0.001 outperforms its counterpart without a weight decay across all metrics, whereas ConvBERT and XLNet had degraded results in all metrics. However, it is important to mention that this is only the case when a cosine learning rate scheduler is applied on DistilBigBird, as DistilBigBird with a linear rate scheduler still outperforms in all aspects a DistilBigbird with a cosine learning rate scheduler and a weight decay of 0.001. Consequently, we chose to drop weight decay entirely from this point onwards.

7.2 Unprocessed Dataset’s Results

This is the best result for the unprocessed dataset after trying out multiple hyperparameters: batch sizes of 4, 8, 16, and 32; learning rates of 1.00e-05, 3.00e-05, and 5.00e-05; learning rate schedulers of cosine and linear form; and weight decay values of 0 and 0.001. In the end, we settled on the following hyperparameters for the unprocessed dataset: a learning rate value of 3.00e-05, a batch size of 8 a weight decay value of 0, a linear learning rate scheduler for DistilBigBird and ConvBERT, and a cosine scheduler for XLNet. The results of training with these hyperparameters are shown in Table 11.

Table 11: Results of training the unprocessed dataset using the best selection of hyperparameters we have found thus far.

Models	F1 Micro	Average Accuracy	F1 Macro	F1 Weighted	F1 Samples
DistilBigBird	64%	62%	59%	63%	60%
ConvBERT	66%	65%	62%	65%	62%
XLNet	66%	64%	62%	65%	62%
Averaged Model	68%	67%	63%	66%	64%

7.3 Applying Preprocessing

We attempted applying a couple of preprocessing steps on the dataset, namely lower-casing the text, expanding contractions, and removing emoticons, URLs, and special characters. Applying the previously mentioned preprocessing operations has resulted in records with empty strings, which are considered useless in training our model, so we had to discard them. Fortunately, the number of records that had to be discarded was insignificant. However, it’s clear to see that preprocessing the dataset degraded the results in comparison to Table 11. This can be explained by the fact that Transformer models are meant to handle natural text out of the box with their own tokenizers.

Table 12: Results of the preprocessed dataset using the best selection of hyperparameters we have found thus far.

Models	F1 Micro	Average Accuracy	F1 Macro	F1 Weighted	F1 Samples
DistilBigBird	61%	57%	53%	58%	57%
ConvBERT	58%	56%	52%	56%	54%
XLNet	57%	57%	48%	53%	52%
Averaged Model	61%	58%	51%	56%	57%

From the comparison of the results of Tables 12 and 11, we conclude that preprocessing is a net negative in our case. However, it’s important to note that the ”totality” of these processes

produced a net negative, and doing each of these in isolation and in different combinations could prove otherwise useful. We will therefore use the dataset without preprocessing in the training from this point onwards.

7.4 Applying Stemming

As seen in Table 13, after using stemming on our unprocessed dataset, the results worsened in relation to Table 11 and this can be explained by the fact that stemming words deprives them of their contextual meaning. Words like "change" and "changing" will be transformed into "chang". Given that Transformer models are relatively performant in "understanding" context, it's only natural such a process would corrupt that context. The insignificance of the change in results however may be explained away by the fact this task may not require detailed contextual indicators like a grammar-checking task or a text-generation task would. It may be that words devoid of context are still viable correlates of personality traits.

Table 13: Results of the stemmed unprocessed dataset using the best selection of hyperparameters we have found thus far.

Models	F1 Micro	Average Accuracy	F1 Macro	F1 Weighted	F1 Samples
DistilBigBird	62%	61%	56%	60%	58%
ConvBERT	65%	63%	60%	63%	61%
XLNet	66%	62%	60%	64%	62%
Averaged Model	66%	64%	60%	64%	62%

From the comparison of the results of Tables 13 and 11, we conclude that stemming is a net negative in our case. We will therefore use the dataset without stemming in the training from this point onwards.

7.5 Applying Lemmatization

Again, as seen in Table 14, lemmatizing the unprocessed dataset degrades the performance in relation to both Table 13 and Table 11, except in the case of the averaged model where lemmatization slightly outperforms stemming (Table 13). Lemmatization does the exact same thing as stemming but it has the benefit of having its resulting words be in the model's vocabulary, unlike "chang" which is not an understood word, however it still adversely affects context similarly to stemming.

Table 14: Results of the lemmatized unprocessed dataset using the best selection of hyperparameters we have found thus far.

Models	F1 Micro	Average Accuracy	F1 Macro	F1 Weighted	F1 Samples
DistilBigBird	61%	61%	58%	61%	57%
ConvBERT	64%	64%	60%	63%	60%
XLNet	64%	64%	60%	63%	60%
Averaged Model	66%	66%	61%	65%	62%

From the comparison of the results of Tables 14 and Table 11, we conclude that lemmatization is a net negative in our case. We will drop lemmatization from our dataset from this point onwards.

7.6 Applying Stop-words Removal

We attempted applying stop-words removal, and as we suspected, that was a net negative as shown when comparing Table 15 with Table 11. In the same vein as stemming and lemmatization, the explanation of the degradation of the results is probably that the removal of stop-words corrupts context, which is something Transformer models are designed to interact with.

Table 15: Results after removing stop-words from the dataset and using the best selection of hyperparameters we have found thus far.

Models	F1 Micro	Average Accuracy	F1 Macro	F1 Weighted	F1 Samples
DistilBigBird	63%	61%	57%	61%	59%
ConvBERT	64%	63%	57%	61%	60%
XLNet	56%	60%	31%	39%	53%
Averaged Model	63%	63%	53%	58%	58%

From the comparison of the results of Tables 15 and Table 11, we conclude that stop-word removal is a net negative in our case. We will drop stop-word removal from our dataset from this point onwards.

7.7 Applying NLP Features

In this section we added NLP features, namely sentiment analysis and NRC features, to our dataset in the hopes of bettering performance. We did this by concatenating the values of these features with the text. For example, if a status message is "Hello, I'm from Saudi Arabia!", after concatenation it will be "The following text: Hello, I'm from Saudi Arabia! has the following sentiment: positive, and the following emotions: anticipation, joy, trust." We did this so that the Transformer models can read these features in sequential token form, something it is provably good at.

7.7.1 Applying Sentiment Analysis

Firstly, we concatenated the sentiments with the text, because we thought that sentiments could be good correlates of personality traits, especially Neuroticism and Agreeableness. The results with the sentiments however were slightly worse than without, as can be seen when comparing Table 16 to Table 11. The explanation could be that sentiments acted as noise during training.

Table 16: Results of training the models on the unprocessed dataset with sentiment analysis using the best selection of hyperparameters we have found thus far.

Models	F1 Micro	Average Accuracy	F1 Macro	F1 Weighted	F1 Samples
DistilBigBird	63%	61%	58%	62%	59%
ConvBERT	66%	64%	62%	65%	62%
XLNet	65%	63%	60%	64%	61%
Averaged Model	67%	66%	62%	66%	64%

From the comparison of the results of Tables 16 and Table 11, we conclude that solely adding sentiment analysis to the unprocessed dataset is a net negative, therefore we will try pairing it with NRC in section 7.7.3 to see if it holds up then.

7.7.2 Applying The NRC Emotion Lexicon

Secondly, in the same way as before, we tried concatenating NRC features with the text. Although yielding better results than the act of solely concatenating sentiments (Table 16), it still performs worse than the unprocessed dataset in Table 11 across all metrics in DistilBigBird and XLNet, except for ConvBERT which showed an increase in results across all metrics except average accuracy, which had a slight decrease.

Table 17: Results of unprocessed dataset with NRC using the best selection of hyperparameters we have found thus far.

Models	F1 Micro	Average Accuracy	F1 Macro	F1 Weighted	F1 Samples
DistilBigBird	62%	61%	58%	61%	58%
ConvBERT	67%	64%	63%	66%	64%
XLNet	63%	62%	51%	57%	58%
Averaged Model	68%	66%	62%	66%	64%

From the comparison of the results of Tables 17 and Table 11, we conclude that solely adding NRC to the unprocessed dataset is a net negative, therefore we will try pairing it with sentiment analysis in the next section to see if it holds up then.

7.7.3 Applying The NRC Emotion Lexicon With Sentiment Analysis

Thirdly, we tried concatenating both NRC and sentiment analysis features with the text. However, to our surprise, the results notably degraded across virtually all metrics, especially in the case of ConvBERT which took a relatively massive drop.

Table 18: Results of unprocessed dataset with NRC and Sentiment Analysis using $3e-5$ as learning rate and batch size of 8.

Models	F1 Micro	Average Accuracy	F1 Macro	F1 Weighted	F1 Samples
DistilBigBird	63%	61%	56%	60%	59%
ConvBERT	56%	60%	31%	39%	53%
XLNet	65%	62%	60%	63%	61%
Averaged Model	64%	63%	55%	60%	59%

From the comparison of the results of Table 18 and Table 11, we conclude that jointly adding NRC with sentiment analysis to the unprocessed dataset is a net negative, therefore we will drop the combination of these two features from our dataset from this point onwards.

7.8 Applying Dataset Undersampling

In order to somewhat combat the biased distribution of the traits in the dataset, we attempted undersampling the trait Openness to get the following distribution as shown in Figure 37. This is to be contrasted with the distribution in Figure 32. Undersampling Openness naturally changes the entire distribution since this is a multi-label classification problem.

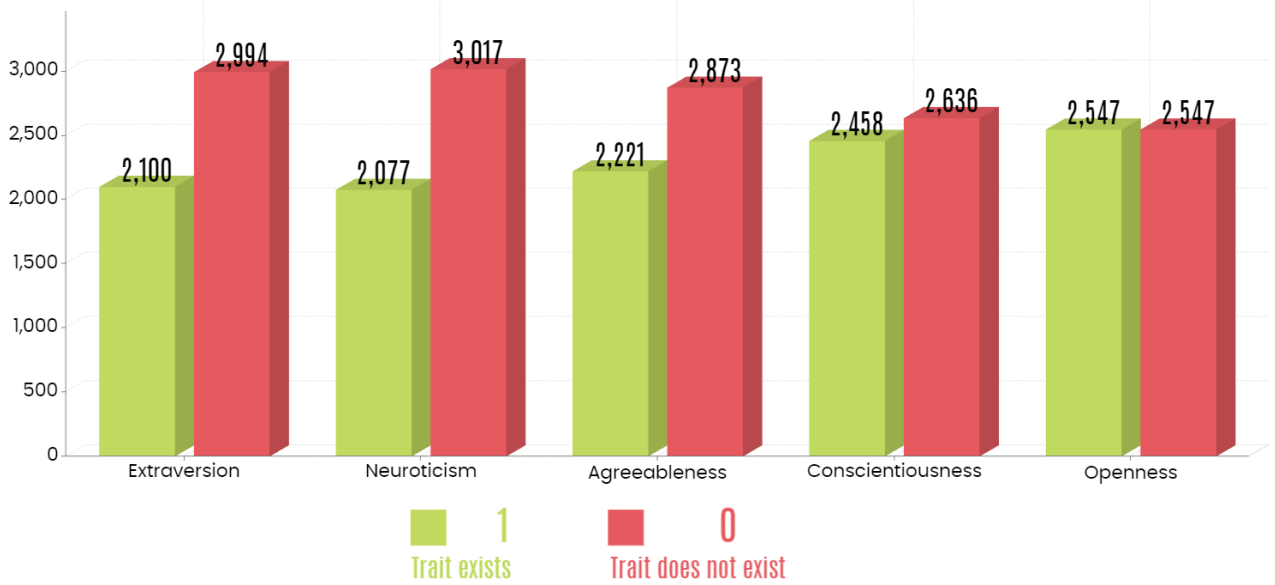


Figure 37: Undersampled class balance in the dataset.

Table 19 shows the results with the best hyperparameters we have found before undersampling, whereas Table 20 shows the results with the same exact hyperparameters after undersampling. It's clear that undersampling was a net negative in terms of performance, however, the Conscientiousness trait was able to improve in F1 and recall after undersampling. Given how undersampling was a net negative, we will go on without it.

Table 19: Results for training the averaged model without undersampling.

Trait	F1	Accuracy	Recall	Precision	Support (Y)	Support (N)
Extraversion	54%	64%	49%	60%	1280	1696
Neuroticism	47%	65%	40%	55%	1123	1853
Agreeableness	69%	64%	76%	64%	1584	1392
Conscientiousness	60%	64%	60%	61%	1375	1601
Openness	86%	76%	95%	77%	2186	790
Average	68%	67%	69%	66%	7548	7332

Table 20: Results for training the averaged model with undersampling.

Trait	F1	Accuracy	Recall	Precision	Support (Y)	Support (N)
Extraversion	53%	64%	51%	54%	602	927
Neuroticism	36%	63%	25%	65%	642	887
Agreeableness	55%	63%	55%	54%	621	908
Conscientiousness	63%	60%	70%	57%	738	791
Openness	65%	61%	75%	57%	741	788
Average	56%	62%	56%	57%	3344	4301

7.9 Best Model Selection

The best model we have found is shown in Table 21 and it has the same hyperparameters in Table 11, but with a learning rate of $2.00\text{e-}05$ in the case of XLNet. Specifically, it has the following hyperparameters and processing steps (all unmentioned are default):

Batch size: 8

Learning rate: $3.00\text{e-}05$ for DistilBigBird and ConvBERT, but $2.00\text{e-}05$ for XLNet.

Weight decay: 0

Learning rate scheduler: "linear" for DistilBigBird and ConvBERT, but "cosine" for XLNet.

Maximum sequence length: Default from each model's tokenizer but with dynamic padding.

Model averaging: averaging on the normalized logits from each model before sigmoid.

Table 21: Results of training the unprocessed dataset using the best selection of hyperparameters we have found.

Models	F1 Micro	Average Accuracy	F1 Macro	F1 Weighted	F1 Samples
DistilBigBird	64%	63%	60%	64%	61%
ConvBERT	67%	65%	63%	66%	64%
XLNet	66%	65%	62%	65%	63%
Averaged Model	69%	68%	65%	68%	66%

Table 22: Final summary of the best model.

Trait	Metrics	DistilBigBird	ConvBERT	XLNet	Averaged Model
Openness	F1	83%	84%	85%	86%
	Accuracy	73%	75%	76%	77%
	Recall	91%	90%	92%	96%
	Precision	76%	79%	79%	79%
	Specificity	19%	32%	29%	25%
Agreeableness	F1	62%	66%	65%	68%
	Accuracy	58%	61%	61%	63%
	Recall	63%	71%	68%	74%
	Precision	60%	62%	62%	63%
	Specificity	53%	50%	53%	51%
Conscientiousness	F1	54%	60%	62%	62%
	Accuracy	60%	65%	62%	66%
	Recall	51%	58%	69%	62%
	Precision	57%	62%	57%	63%
	Specificity	69%	70%	56%	70%
Extraversion	F1	56%	60%	52%	60%
	Accuracy	60%	63%	62%	65%
	Recall	61%	68%	49%	63%
	Precision	52%	54%	55%	57%
	Specificity	59%	59%	71%	66%
Neuroticism	F1	48%	46%	46%	48%
	Accuracy	62%	63%	64%	67%
	Recall	46%	42%	40%	40%
	Precision	50%	52%	53%	58%
	Specificity	72%	76%	79%	83%
Averaged	F1	64%	67%	66%	69%
	Accuracy	63%	65%	65%	68%
	Recall	66%	70%	68%	71%
	Precision	62%	65%	65%	67%
	Specificity	54%	57%	58%	59%

Chapter 8: Conclusions and Future Works

Understanding personalities is instrumental in understanding and predicting future behaviors. As such, it's well-advised to make use of whatever resources that further our understanding of personalities, especially if such understanding is carried out automatically, as in the case of our research wherein both textual data and personality test results are fed to deep learning models that output author personality classifications based on The Big Five personality framework. This task hopes to satisfy use-cases for which traditional survey-based personality tests are insufficient.

In the first phase of the project, we gave a brief introduction to our problem and then a background to the relevant subjects to our work, such as what a personality is and different personality measures. We also explained natural language processing (NLP), deep learning and its various techniques, supervised machine learning and its various techniques, and ensemble learning. We proposed a model, which can predict the personalities of text authors using the Big Five personality measure; however, before we could accomplish that, we needed to figure out which techniques to employ, so we conducted a review of the literature that compared different approaches to the same problem. The architecture used in our proposed system goes through different stages starting with preprocessing the myPersonality project dataset, then inserting the preprocessed dataset into three state-of-the-art transformers: XLNet, BigBird, and ConvBERT. While also using Sentiment analysis, TF-IGM, and NRC Emotion Lexicon. The results will be then inserted into feed-forward neural networks, so then we compute the average for the final prediction. To evaluate our model we used performance measures like precision, recall, accuracy, and F1 score.

In the second phase of the project, we implemented the proposed model and tested it using standard performance measures. We tried out different hyperparameter settings and techniques, and evaluated multiple of their combinations. We found out that our proposed model was in fact worse than some of the other combinations we have tried. In particular, the addition of NRC and Sentiment Analysis yielded no net positive to the best model we have created. Contrarily, the addition of model averaging made a net positive. We also explored the dataset more and uncovered more of its flaws.

For future works, we recommend researchers create a far bigger and more balanced dataset, and experiment with models with bigger bases, use different types of averaging such as weighted averaging to make it so that if one model often misclassifies a specific trait they get a smaller weighted vote, and vice-versa.

References:

- [1] Wiegmann, M., Stein, B. Potthast, M. (2019). "Overview of the Celebrity Profiling Task at PAN 2019." CLEF.
- [2] Yang, H.C., Huang, Z.R.: Mining personality traits from social messages for game recommender systems. *Knowledge-Based Systems* 165, 157–168 (2019)
- [3] Matz, S. and Hirsh, J.B. (2020). Marketing and Personality. In *The Wiley Encyclopedia of Personality and Individual Differences* (eds B.J. Carducci, C.S. Nave, J.S. Mio and R.E. Riggio).
- [4] Mehta, Y., Majumder, N., Gelbukh, A. et al. Recent trends in deep learning based personality detection. *Artif Intell Rev* 53, 2313–2339 (2020). <https://doi.org/10.1007/s10462-019-09770-z>
- [5] Liem, C.C., Langer, M., Demetriou, A., Hiemstra, A.M., Wicaksana, A.S., Born, M.P., Konig, C.J.: Psychology meets machine learning: Interdisciplinary perspectives on algorithmic job candidate screening. In: *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pp. 197–253. Springer (2018)
- [6] Min Yang, Kam-Pui Chow. Authorship Attribution for Forensic Investigation with Thousands of Authors. 29th IFIP International Information Security Conference (SEC), Jun 2014, Marrakech, Morocco. pp.339-350.
- [7] Stuart J. Russell, Peter Norvig (2010) *Artificial Intelligence: A Modern Approach*, Third Edition, Prentice Hall ISBN 9780136042594.
- [8] Alakh, Sethi. "Supervised Learning vs. Unsupervised Learning – A Quick Guide for Beginners." *Analytics Vidhya*, 6 April 2020, <https://www.analyticsvidhya.com/blog/2020/04/supervised-learning-unsupervised-learning/>.
- [9] Feist, J., Feist, G. J. (2009). *Theories of personality* (7th ed.), Boston: McGraw-Hill.
- [10] Burkov, A., Munar, A., Handlin, C. (2022). *The Hundred-Page Machine Learning Book*. Andriy Burkov.
- [11] Supervised Learning. 2020. Image. <https://www.tutorialandexample.com/wp-content/uploads/2020/11/Supervised-Machine-Learning-1.png>.
- [12] Brownlee, J., 2022. 4 Types of Classification Tasks in Machine Learning. [online] *Machine Learning Mastery*. Available at: <https://machinelearningmastery.com/types-of-classification-in-machine-learning/> [Accessed 7 March 2022].
- [13] Mohamed, Aly (2005). "Survey on multiclass classification methods" (PDF). Technical Report, Caltech.
- [14] Feist, J., Feist, G. J. (2009). *Theories of personality* (7th ed.), Boston: McGraw-Hill.
- [15] O. P. John, S. Srivastava, "The big five trait taxonomy: History, measurement, and theoretical perspectives" in *Handbook of Personality: Theory and Research*, L. A. Pervin, O. P. John, Eds. (Guildford Press, New York, NY, 1999), pp. 102–138.
- [16] Power RA, Pluess M. Heritability estimates of the Big Five personality traits based on common genetic variants. *Transl Psychiatry*. 2015;5:e604.
- [17] Gregory J. Boyle (3-1-1995). *Myers-Briggs Type Indicator (MBTI): Some Psychometric Limitations*

- [18] Butcher, J. N., Bubany, S., Mason, S. N. (2013). Assessment of personality and psychopathology with self-report inventories. In K. F. Geisinger, B. A. Bracken, J. F. Carlson, J.-I. C. Hansen, N. R. Kuncel, S. P. Reise, M. C. Rodriguez (Eds.), *APA handbook of testing and assessment in psychology*, Vol. 2. Testing and assessment in clinical and counseling psychology (pp. 171–192). American Psychological Association. <https://doi.org/10.1037/14048-011>
- [19] Smith SR. Projective Assessment Techniques. In: Goldstein S., Naglieri J.A, eds., *Encyclopedia of Child Behavior and Development*. Springer, Boston, MA; 2011. doi:10.1007/978-0-387-79061-9
- [20] Moyle P, Hackston J. Personality assessment for employee development: Ivory tower or real world?. *J Pers Assess*. 2018;100(5):507–517. doi:10.1080/00223891.2018.1481078
- [21] Iudici A, Salvini A, Faccio E, Castelnuovo G. The clinical assessment in the legal field: An empirical study of bias and limitations in forensic expertise. *Front Psychol*. 2015;6:1831. doi:10.3389/fpsyg.2015.01831
- [22] Bäckström M, Björklund F. Social desirability in personality inventories: symptoms, diagnosis and prescribed cure. *Scand J Psychol*. 2013;54(2):152–159. doi:10.1111/sjop.12015
- [23] Vazire S, Carlson EN. Self-knowledge of personality: Do people know themselves? *Soc Personal Psychol Compass*. 2010;4(8):605–620. doi:10.1111/j.1751-9004.2010.00280.x
- [24] McCrae RR, Kurtz JE, Yamagata S, Terracciano A. Internal consistency, retest reliability, and their implications for personality scale validity. *Pers Soc Psychol Rev*. 2011;15(1):28– 50. doi:10.1177/1088868310366253
- [25] Cherry, K. What Is a Personality Test. Updated (14/2/2022), Medically reviewed by Amy Morin, LCSW
- [26] Lim, A. (2020, June 15). 'The big five personality traits'.Simply Psychology. <https://www.simplypsychology.org/big-five-personality.html>
- [27] Personality Type Based on Myers-Briggs Type Indicator with Text Posting Style by using Traditional and Deep Learning - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/A-chart-with-descriptions-of-each-Myers-Briggs\protect\@normalcr\relaxality-type-24_fig1_358145283 [accessed 23 Mar, 2022]
- [28] Sharma, Eishita Mahajan, Rhea Mahajan, Remia Mansotra, Vibhakar. (2021). Automated Personality Prediction of Social Media Users: A Decade Review. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*. 12. 5225-5237
- [29] Xue, D., Wu, L., Hong, Z. et al. Deep learning-based personality recognition from text posts of online social networks. *Appl Intell* 48, 4232–4246 (2018). <https://doi.org/10.1007/s10489-018-1212-4>
- [30] Jerome H. Friedman "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, Ann. Statist. 29(5), 1189–1232, (October 2001)
- [31] myPersonality.org. (2007). myPersonality. Retrieved March 20, 2022, from <https://sites.google.com/michalkosinski.com/mypersonality>
- [32] M. M. Tadesse, H. Lin, B. Xu and L. Yang, "Personality Predictions Based on User Behavior on the Facebook Social Media Platform," in *IEEE Access*, vol. 6, pp. 61959–61969, 2018, doi: 10.1109/ACCESS.2018.2876502.

- [33] Rizwan Iqbal, H., Adnan Ashraf, M., Adeel Nawab, R. M. (2015). Predicting an author's demographics from text using Topic Modeling approach. Predicting an Author's Demographics from Text Using Topic Modeling Approach. <http://ceur-ws.org/Vol-1391/75-CR.pdf>
- [34] anaN, N., Thri, V. (2018). Performance and Classification Evaluation of J48 Algorithm and Kendall's Based J48 Algorithm (KNJ48). International Journal of Computer Trends and Technology, 59(2), 73–80. <https://doi.org/10.14445/22312803/ijctt-v59p112>
- [35] Kosan, M. A., Karacan, H., Urgan, B. A. (2022). Predicting personality traits with semantic structures and LSTM-based neural networks. Alexandria Engineering Journal, 61(10), 8007–8025. <https://doi.org/10.1016/j.aej.2022.01.050>
- [36] Kunte, A. V., Panicker, S. (2019). Using textual data for Personality Prediction:A Machine Learning Approach. 2019 4th International Conference on Information Systems and Computer Networks (ISCON). <https://doi.org/10.1109/iscon47742.2019.9036220>
- [37] Khan, A., Ahmad, H., Zubair, M., Khan, F., Arif, A., Ali, H. (2020). Personality Classification from Online Text using Machine Learning Approach. International Journal of Advanced Computer Science and Applications, 11
- [38] Hassanein, M., Hussein, W., Rady, S., Gharib, T. F. (2018). Predicting Personality Traits from Social Media using Text Semantics. 2018 13th International Conference on Computer Engineering and Systems (ICCES). <https://doi.org/10.1109/icces.2018.8639408>
- [39] R. Akhtar, D. Winsborough, U. Ort, A. Johnson, and T. Chamorro-Premuzic, "Detecting the dark side of personality using social media status updates," Pers. Individ. Dif., vol. 132, no. October 2017, pp. 90–97, 2018.
- [40] P. D. Turney and P. Pantel, "From Frequency to Meaning Vector Space Models of Semantics," J. Artif. Intell. Res., vol. 37, pp. 141–188, 2010.
- [41] B. T. McInnes, T. Pedersen, Y. Liu, G. B. Melton, and S. V. Pakhomov, "Knowledge-based Method for Determining the Meaning of Ambiguous-Biomedical Terms Using Information Content Measures of Similarity," AMIA Annu. Symp. Proc., vol. 2011, p. 895, 2011.
- [42] M. Palmer and Z. Wu, "VERB SEMANTICS AND LEXICAL Zhibiao Wu," Proc. ACL, pp. 133–138, 1994
- [43] H. Zhang, C. L. Giles, H. C. Foley, and J. Yen. Probabilistic community discovery using hierarchical latent Gaussian mixture model. In AAAI'07: Proceedings of the 22nd National Conference on Artificial Intelligence, pages 663–668, (2007).
- [44] Christian, H., Suhartono, D., Chowanda, A., Zamli, K. Z. (2021b). Text based personality prediction from multiple social media data sources using pre-trained language model and model averaging. Journal of Big Data, 8(1). <https://doi.org/10.1186/s40537-021-00459-1>
- [45] evlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina (11 October 2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". arXiv:1810.04805v2 [cs.CL].
- [46] "Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing". Google AI Blog.
- [47] Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M,

- Zettlemoyer L, Stoyanov V. RoBERTa: A robustly optimized BERT pre-training approach. ArXiv; 2019. 1 <https://arxiv.org/abs/1907.11692v1>
- [48] Yang Z, Dai Z, Yang Y, Carbonell J, Salakhutdinov R, Le QV. XLNet: generalized autoregressive pretraining for language understanding. ArXiv, NeurIPS; 2019. pp. 1–18.
- [49] Bin Tareaf R, Berger P, Hennig P, Meinel C. Cross-platform personality exploration system for online social networks: Facebook vs. Twitter Web Intell. 2020;18(1):35–51. <https://doi.org/10.3233/WEB-200427>.
- [50] Chen, K., Zhang, Z., Long, J., Zhang, H. (2016). Turning from TF-IDF to TF-IGM for term weighting in text classification. Expert Systems with Applications, 66, 245–260. <https://doi.org/10.1016/j.eswa.2016.09.009>
- [51] Mohammad, S. M., Turney, P. D. (2012). CROWDSOURCING A WORD-EMOTION ASSOCIATION LEXICON. Computational Intelligence, 29(3), 436–465. <https://doi.org/10.1111/j.1467-8640.2012.00460.x>
- [52] O. P. John, S. Srivastava, “The big five trait taxonomy: History, measurement, and theoretical perspectives” in Handbook of Personality: Theory and Research, L. A. Pervin, O. P. John, Eds. (Guildford Press, New York, NY, 1999), pp. 102–138.
- [53] Schuster, Mike Paliwal, Kuldip. (1997). Bidirectional recurrent neural networks. Signal Processing, IEEE Transactions on. 45. 2673 - 2681. 10.1109/78.650093.
- [54] Hochreiter, Sepp Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.
- [55] Tandra, T., Hendro, Suhartono, D., Wongso, R., Prasetyo, 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>
- [56] Apply Magic Sauce - Prediction API. (2014). Apply Magic Sauce. Retrieved March 23, 2022, from <https://applymagicsauce.com/demo>
- [57] Cho, Kyunghyun Merrienboer, Bart Gulcehre, Caglar Bougares, Fethi Schwenk, Holger Bengio, Y.. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. 10.3115/v1/D14-1179.
- [58] Bengio, Y. Lecun, Yann. (1997). Convolutional Networks for Images, Speech, and Time-Series.
- [59] Chollet Francois. (2018). Deep learning with python. Manning Publications Co.
- [60] Neural Network. (2021). [Illustration]. https://1.cms.s81c.com/sites/default/files/2021-01-06/ICLH_Diagram_Batch_01_03-DeepNeuralNetwork-WHITEBG.png
- [61] Rumelhart, D., Hinton, G. Williams, R. Learning representations by back-propagating errors. Nature 323, 533–536 (1986). <https://doi.org/10.1038/323533a0>
- [62] Hochreiter, Sepp Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.
- [63] Cho, Kyunghyun, et al. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. 2014. DOI.org (Datacite), <https://doi.org/10.48550/ARXIV.1409.1259>.
- [64] (n.d.). LSTM and GRU architectures. Retrieved March 25, 2022, from

- https://miro.medium.com/max/1400/1*yBXV9o5q7L_CvY7quJt3WQ.png.
- [65] Vaswani, Ashish, et al. Attention Is All You Need. 2017. DOI.org (Datacite), <https://doi.org/10.48550/ARXIV.1706.03762>.
- [66] CNN. (2020). [Illustration]. <https://editor.analyticsvidhya.com/uploads/90650dnn2.jpeg>
- [67] RNN. (2020). [Illustration]. https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/assets_-LvBP1svpACTB1R1x_U4_-LwEQnQw8wHRB6_2zYtG_-LwEZT8zd07mLDuaQZwy_image-1.png
- [68] Kumar, A., Jain, M. (2020). Ensemble Learning for AI Developers. Apress.
- [69] Zhang, C., Ma, Y. (2012). Ensemble Machine Learning: Methods and Applications (2012th ed.). Springer.
- [70] Zhou, Z. (2012). Ensemble Methods: Foundations and Algorithms (Chapman Hall/CRC Machine Learning Pattern Recognition) (1st ed.).
- [71] Rieger, Steven A.; Muraleedharan, Rajani; Ramachandran, Ravi P. (2014). Speech based emotion recognition using spectral feature extraction and an ensemble of kNN classifiers. Proceedings of the 9th International Symposium on Chinese Spoken Language Processing, ISCSLP 2014. pp. 589–593. doi:10.1109/ISCSLP.2014.6936711. ISBN 978-1-4799-4219-0. S2CID 31370450.)
- [72] Ocasto, Michael E.; Wang, Ke; Keromytis, Angeles D.; Salvatore, J. Stolfo (2005). FLIPS: Hybrid Adaptive Intrusion Prevention. Recent Advances in Intrusion Detection. Lecture Notes in Computer Science. Vol. 3858. pp. 82–101. CiteSeerX 10.1.1.60.3798. doi:10.1007/116638125.ISBN978-3-540-31778-4.
- [73] Saini, A.N.W.A.R.(2021, January28).BaggingEnsemble[Graph].
- [74] Rocca, J.(2019, April23).BoostingEnsemble[Graph].
- [75] Brownlee, J. (2021, April 19). Stacking Ensemble [Graph]. cyan<https://machinelearningmastery.com/wp-content/uploads/2020/11/Stacking-Ensemble.png>
- [76] Gaikwad, D., Thool, R. C. (2015). Intrusion Detection System Using Bagging Ensemble Method of Machine Learning. 2015 International Conference on Computing Communication Control and Automation. cyan<https://doi.org/10.1109/iccube.2015.61>
- [77] Maclin, R., Opitz, D. (1997). An Empirical Evaluation of Bagging and Boosting. An Empirical Evaluation of Bagging and Boosting. cyan<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.6964&rep=rep1&type=pdf>
- [78] Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., Herrera, F. (2012). A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(4), 463–484. cyan<https://doi.org/10.1109/tsmcc.2011.2161285>
- [79] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” J. Comput. Syst. Sci., vol. 55, no. 1, pp. 119–139, 1997.
- [80] Wang, G., Hao, J., Ma, J., Jiang, H. (2011). A comparative assessment of ensemble learning for credit scoring. Expert Systems with Applications, 38(1), 223–230. cyan<https://doi.org/10.1016/j.eswa.2010.06.048>

- [81] J. Yan and S. Han, "Classifying imbalanced data sets by a novel re-sample and cost-sensitive stacked generalization method," *Mathematical Problems in Engineering*, vol. 2018, Article ID 5036710, 13 pages, 2018.
- [82] Mayo, M. (2018, October 17). The Main Approaches to Natural Language Processing Tasks. KDnuggets. [cyanhttps://www.kdnuggets.com/2018/10/main-approaches-natural-language-processingtasks.html](https://www.kdnuggets.com/2018/10/main-approaches-natural-language-processingtasks.html)
- [83] Goldberg, Y., Hirst, G. (2017). *Neural Network Methods in Natural Language Processing*. Macmillan Publishers
- [84] Bird, S., Klein, E., Loper, E. (2009). *Natural Language Processing with Python*. Van Duuren Media.
- [85] Aditya Jain, Gandhar Kulkarni, Vraj Shah, (2018). *Natural Language Processing*. *International Journal of Computer Sciences and Engineering*, 6(1), 161-167.
- [86] Nadkarni P.M., Ohno-Machado L., Chapman W.W. *Natural language processing: an introduction*. *J. Am. Med. Inform. Assoc.* 2011;18:544–551. - PMC - PubMed
- [87] Klein D. CS 294e5: *Statistical Natural Language Processing*. 2005. <http://www.cs.berkeley.edu/wklein/cs294-5>
- [88] Manning, C., Schutze, H. (1999). *Foundations of Statistical Natural Language Processing*. Amsterdam University Press.
- [89] Joshi, A. K. (1991). *Natural Language Processing*. *Science*, 253(5025), 1242–1249. [cyanhttps://doi.org/10.1126/science.253.5025.1242](https://doi.org/10.1126/science.253.5025.1242)
- [90] Vajjala, S., Majumder, B., Gupta, A., Surana, H. (2020). *Practical Natural Language Processing*. Van Duuren Media.
- [91] NLTK : Natural Language Toolkit. (n.d.-b). *Natural Language Toolkit*. [cyanhttps://www.nltk.org/](https://www.nltk.org/)
- [92] Marcus, M. (1995). New trends in natural language processing: statistical natural language processing. *Proceedings of the National Academy of Sciences*, 92(22), 10052–10059. [cyanhttps://doi.org/10.1073/pnas.92.22.10052](https://doi.org/10.1073/pnas.92.22.10052)
- [93] Hirschberg, J., Manning, C. D. (2015). Advances in natural language processing. *Science*, 349(6245), 261–266. [cyanhttps://doi.org/10.1126/science.aaa8685](https://doi.org/10.1126/science.aaa8685)
- [94] displaCy Named Entity Visualizer . (n.d.). *Explosion*. <https://explosion.ai/demos/displacy-ent>
- [95] NLTK :: Sample usage for relextract. (n.d.). *Natural Language Toolkit*. [cyanhttps://www.nltk.org/howto/relextract.html](https://www.nltk.org/howto/relextract.html)
- [96] Clark, Kevin Manning, Christopher. (2016). Improving Coreference Resolution by Learning Entity- Level Distributed Representations. 643-653. 10.18653/v1/P16-1061.
- [97] Manning, C. D., Clark, K., Hewitt, J., Khandelwal, U., Levy, O. (2020). Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48), 30046–30054. [cyanhttps://doi.org/10.1073/pnas.1907367117](https://doi.org/10.1073/pnas.1907367117)
- [98] hugging face – The AI community building the future. (n.d.). huggingface. [cyanhttps://huggingface.co/](https://huggingface.co/)
- [99] spaCy · Industrial-strength Natural Language Processing in Python. (n.d.). spaCy. [cyanhttps://spacy.io/](https://spacy.io/)

- [100] Neural Coreference – Hugging Face. (n.d.). Huggingface. <https://huggingface.co/coref/>
- [101] Burkov, A. (2019b). *The Hundred-Page Machine Learning Book*. Andriy Burkov.
- [102] Bengfort, B., Bilbro, R., Ojeda, T. (2018). *Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning* (1st ed.). O'Reilly Media.
- [103] Padarian, J., Fuentes, I. (2019). Word embeddings for application in geosciences: development, evaluation, and examples of soil-related concepts. *SOIL*, 5(2), 177–187. <https://doi.org/10.5194/soil-5-177-2019>
- [104] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2nd ed.). O'Reilly Media.
- [105] SPARCK JONES, K. (1972). A STATISTICAL INTERPRETATION OF TERM SPECIFICITY AND ITS APPLICATION IN RETRIEVAL. *Journal of Documentation*, 28(1), 11–21. <https://doi.org/10.1108/eb026526>
- [106] Chen, K., Zhang, Z., Long, J., Zhang, H. (2016). Turning from TF-IDF to TF-IGM for term weighting in text classification. *Expert Systems with Applications*, 66, 245–260. <https://doi.org/10.1016/j.eswa.2016.09.009>
- [107] Mikolov, Tomas Chen, Kai Corrado, G.s Dean, Jeffrey. (2013). Efficient Estimation of Word Representations in Vector Space. *Proceedings of Workshop at ICLR*. 2013.
- [108] Pennington, Jeffrey Socher, Richard Manning, Christopher. (2014). Glove: Global Vectors for Word Representation. *EMNLP*. 14. 1532-1543. [10.3115/v1/D14-1162](https://doi.org/10.3115/v1/D14-1162).
- [109] Chakrabarty, Navoneel. (2019). *A Machine Learning Approach to Comment Toxicity Classification*.
- [110] Jurafsky, D., Martin, J. H. (2009). *Speech and Language Processing*. Prentice Hall.
- [111] Fig. 1. (2019, December 19). [Graph]. Kern, M. L., McCarthy, P. X., Chakrabarty, D., Rizoio, M. A. (2019). Social Media-Predicted Personality Traits and Values Can Help Match People to Their Ideal Jobs. <https://www.pnas.org/cms/10.1073/pnas.1917942116/asset/9ce97abb-fe3c-4b09-909a-b03ef4f6abb9/assets/graphic/pnas.1917942116fig01.jpeg>
- [112] Luhn, H. P. (1957). A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal of Research and Development*, 1(4), 309–317. <https://doi.org/10.1147/rd.14.0309>
- [113] Zen Flowchart. (n.d.). Zen Flowchart. <https://app.zenflowchart.com/>
- [114] Reddy, T. & Bulusu, Vishnu vardhan & Reddy, Vijayapal. (2016). A survey on Authorship Profiling techniques. 11. 3092-3102.
- [115] Myers, I. B., McCaulley, M. H., & Most, R. (1985). *Manual, a guide to the development and use of the Myers-Briggs type indicator*. consulting psychologists press.
- [116] Marcia Carlyn (1977) An Assessment of the Myers-Briggs Type Indicator, *Journal of Personality Assessment*, 41:5, 461-473, DOI: [10.1207/s15327752jpa4105_2](https://doi.org/10.1207/s15327752jpa4105_2)
- [117] Christian, H., Suhartono, D., Chowanda, A. et al. Text based personality prediction from multiple social media data sources using pre-

- trained language model and model averaging. *J Big Data* 8, 68 (2021). <https://doi.org/10.1186/s40537-021-00459-1>
- [118] Zhang, L., Wang, S., & Liu, B. (2018). Deep learning for sentiment analysis: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(4). <https://doi.org/10.1002/widm.1253>
- [119] Chen, K., Zhang, Z., Long, J., & Zhang, H. (2016). Turning from TF-IDF to TF-IGM for term weighting in text classification. *Expert Systems with Applications*, 66, 245–260. doi:10.1016/j.eswa.2016.09.009
- [120] Devlin, Jacob & Chang, Ming-Wei & Lee, Kenton & Toutanova, Kristina. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
- [121] Dai, Zihang & Yang, Zhilin & Yang, Yiming & Carbonell, Jaime & Le, Quoc & Salakhutdinov, Ruslan. (2019). Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. 2978-2988. 10.18653/v1/P19-1285.
- [122] Wang, Alex & Singh, Amanpreet & Michael, Julian & Hill, Felix & Levy, Omer & Bowman, Samuel. (2018). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. 353355. 10.18653/v1/W18-5446
- [123] Rajpurkar, Pranav & Zhang, Jian & Lopyrev, Konstantin & Liang, Percy. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. 2383-2392. 10.18653/v1/D16-1264.
- [124] Lai, Guokun & Xie, Qizhe & Liu, Hanxiao & Yang, Yiming & Hovy, Eduard. (2017). RACE: Large-scale ReAding Comprehension Dataset From Examinations.
- [125] Yelp Dataset. (n.d.). Yelp Dataset. <https://www.yelp.com/dataset>
- [126] IMDB Datasets. (n.d.). IMDB. <https://www.imdb.com/interfaces/>
- [127] The ClueWeb09 Dataset. (n.d.). lemurproject. <https://lemurproject.org/clueweb09/>
- [128] Zaheer, Manzil & Guruganesh, Guru & Dubey, Avinava & Ainslie, Joshua & Alberti, Christopher & Ontanon, Santiago & Pham, Philip & Ravula, Anirudh & Wang, Qifan & Yang, Li & Ahmed, Amr. (2020). Big Bird: Transformers for Longer Sequences.
- [129] Jiang, Zihang & Yu, Weihao & Daquan, Zhou & Chen, Yunpeng & Feng, Jiashi & Yan, Shuicheng. (2020). ConvBERT: Improving BERT with Span-based Dynamic Convolution.
- [130] Clark, Kevin & Luong, Minh-Thang & Le, Quoc & Manning, Christopher. (2020). ELECTRA: Pretraining Text Encoders as Discriminators Rather Than Generators.
- [131] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning* Volume 70, pages 933–941. JMLR. org, 2017.
- [132] Edward Ma. 2019. NLP Augmentation. <https://github.com/makcedward/nlpaug>.
- [133] Patterson, Josh; Gibson, Adam (2017). "Understanding Learning Rates". *Deep Learning : A Practitioner's Approach*. O'Reilly. pp. 258–263. ISBN 978-1-4919-1425-0.
- [134] SajjadAyoubi/distil-bigbird-fa-zwnj · Hugging Face. (n.d.). Retrieved November 7, 2022, from <https://huggingface.co/SajjadAyoubi/distil-bigbird->

fa-zwnj?text=The+goal+of+life+is+.

[135] PyTorch. (n.d.). Retrieved November 7, 2022, from <https://pytorch.org/>

[136] NumPy. (n.d.). Retrieved November 7, 2022, from <https://numpy.org/>

[137] pandas - Python Data Analysis Library. (n.d.). Retrieved November 7, 2022, from <https://pandas.pydata.org/>

[138] Transformers. (n.d.). Retrieved November 7, 2022, from <https://huggingface.co/docs/transformers/main/en/index>

[139] Datasets. (n.d.). Retrieved November 7, 2022, from <https://huggingface.co/docs/datasets/index>

[140] scikit-learn: machine learning in Python — scikit-learn 1.1.3 documentation. (n.d.). Retrieved November 7, 2022, from <https://scikit-learn.org/stable/>

[141] fast-ml. (2021, July 11). PyPI. <https://pypi.org/project/fast-ml/>

[142] Welcome to. (2022, November 1). Python.org. <https://www.python.org/>

[143] Visual Studio Code - Code Editing. Redefined. (2021, November 3). <https://code.visualstudio.com/>

[144] Project Jupyter. (n.d.). Home. Retrieved November 7, 2022, from <https://jupyter.org/>

[145] <https://jupyterlab.readthedocs.io/en/stable/>

[146] <https://jupyter-notebook.readthedocs.io/en/stable/>

[147] Google Colaboratory. (n.d.). Retrieved November 7, 2022, from <https://colab.research.google.com/>

[148] Siegfried Bos and E Chug. Using weight decay to optimize the generalization ability of a perceptron. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pp. 241–246. IEEE, 1996.

[149] Gaillard, F., Murphy, A. Batch size (machine learning). Reference article, Radiopaedia.org. (accessed on 07 Nov 2022)

[150] Murphy, Kevin P. (2012). *Machine Learning: A Probabilistic Perspective*. Cambridge: MIT Press. p. 247. ISBN 978-0-262-01802-9.

[151] Papa, J. (n.d.). *PyTorch Pocket Reference*. O'Reilly Online Learning. <https://www.oreilly.com/library/view/pytorch-pocket-reference/9781492089995/>

[152] Buduma, Nikhil; Locascio, Nicholas (2017). *Fundamentals of Deep Learning : Designing Next-Generation Machine Intelligence Algorithms*. O'Reilly. p. 21. ISBN 978-1-4919-2558-4.

[153] Mohammad, S.M., & Turney, P.D. (2013). CROWDSOURCING A WORD-EMOTION ASSOCIATION LEXICON. *Computational Intelligence*, 29.

[154] Mohammad, S.M., & Turney, P.D. (2010). Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create an Emotion Lexicon. *HLT-NAACL 2010*.