

MSC Computer Science & Digital Technologies

TITLE

The role of Machine Learning in the fight against online Anti-Social Behaviour.

Author

Haiel-Marie H. W. AGBO
MSc Cyber Security with Advance Practice
W18041453
Northumbria University, London

Supervisor

Usman Butt
Associate Dean/Head of Department Computing
Northumbria University, London

2020-2021

Declaration

I declare the following:

1. That the material contained in this dissertation is the result of my own work and that due acknowledgement has been given in the bibliography and references to ALL sources be they printed, electronic or personal.
2. The Word Count of this Dissertation is 15070.
3. That unless this dissertation has been confirmed as confidential, I agree to an entire electronic copy or sections of the dissertation to being placed on the eLearning Portal (Blackboard), if deemed appropriate, to allow future students the opportunity to see examples of past dissertations. I understand that if displayed on eLearning Portal it would be made available for no longer than five years and that students would be able to print off copies or download.
4. I agree to my dissertation being submitted to a plagiarism detection service, where it will be stored in a database and compared against work submitted from this or any other Department or from other institutions using the service.
5. In the event of the service detecting a high degree of similarity between content within the service this will be reported back to my supervisor and second marker, who may decide to undertake further investigation that may ultimately lead to disciplinary actions, should instances of plagiarism be detected.
6. I have read the Northumbria University Policy Statement on Ethics in Research and Consultancy and I confirm that ethical issues have been considered, evaluated, and appropriately addressed in this research.

SIGNED: Haiiel-Marie H.W. AGBO

Date: 29/01/2020

Acknowledgement

I would like to thank the following people, without whom I would not have been able to complete this thesis, and without whom I would not have made it through my master's degree.

First of all, my parents for the mental and financial support provided throughout these two years of my master's degree. I particularly thank them for being patient and understanding about the discipline and sacrifices that were required for the completion of this work.

I would like to thank Professor Hamid Jahankhani for his recommendations and infallible support throughout this process in time of need. Going through the program with him as a program leader was truly a great experience. I would also like to thank my supervisor Usman Butt for all the help and especially for the opportunity he provided me with to grow as a researcher and further improve my writing skills. It is without a doubt thanks to the experience gained that I was able to complete this dissertation.

I would also like to thank all lecturers, tutors, and other staff of the University of Northumbria University London. David Haselkorn states that: "Teaching is the essential profession, the one that makes all professions possible". Thank you all for helping me build a consistent career path.

Finally, I would like to give a special thanks to Amna Abubakar Ahmad for the constant motivation and moral support during the most critical times of this dissertation. Mahatma Gandhi argues that "The best way to find yourself is to lose yourself in the service of others". Thank you for everything and especially for the selfless support.

Abstract

The era of social media first began in 1997 with the introduction of a website named “SixDegree.com”. It was originally designed for users to set up a profile, create connections and send messages within the network. Since then, the concept of social media has increased exponentially giving way to social media giants such as Facebook, Twitter, Instagram, TikTok and many more being used on a daily by people all over the world. However, due to large scale of social media, the difficulty faced by governments to regulate the environment by law and the freedom of speech given by those networks, unacceptable social behaviour such as cyber-bullying, use of hate speech and abusive language, extremism and other terrible behaviours started occurring without any mean to control them. As a result of such behaviour, individuals targeted displayed signs of depression, frustration, and mental trauma. In this dissertation, we propose a method for the identification and classification of abusive language in a real-life situation. To that end, a systematic literature review was done on various pre-processing methods/techniques and machine learning algorithms and classifiers to grasp the way they function with each other and to also understand how these methods were previously used in the context of text classification. Furthermore, after analysing and reviewing existing text classification models, a more detailed text classification process was proposed for this dissertation. Finally, the machine learning model was designed following the proposed text classification process. As a result, a pre-trained classification model was used and embedded into a chat application and showed great results for the identification of offensive language, slang, and others. To summarize, the proposed approach can identify an unacceptable language in conversations using machine learning and classification techniques.

Table of Contents

Declaration.....	2
Acknowledgement.....	3
Abstract.....	4
Table of Contents.....	5
I. Introduction.....	11
1. Background Overview/ Motivation	11
2. Aim.....	12
3. Objectives.....	12
4. Structure of the Dissertation	12
II. Literature Review.....	14
1. Natural Language Processing (NLP)	14
1.1. Syntactic Analysis.....	15
1.1.1. Lemmatization.....	15
1.1.2. Stemming	17
1.1.3. Morphological Segmentation.....	17
1.1.4. Word/Text Segmentation	18
1.1.5. Part-of-Speech Tagging (POS)	19
1.1.6. Parsing.....	21
1.2. Semantic Analysis.....	23
1.2.1. Named Entity Recognition (NER).....	23
1.2.2. Word Sense Disambiguation (WSD)	24
1.2.3. Natural Language Generation (NLG)	27
2. Text Classification	29
2.1. Text Classification Process	30
2.1.1. Document Collection	31
2.1.2. Pre-Processing.....	31
2.1.3. Indexing.....	31
2.1.4. Feature Extraction/Selection (FE/FS).....	33
2.1.5. Classification.....	36
2.1.5.1. Rocchio Algorithm.....	37
2.1.5.2. Boosting	39
2.1.5.3. Bagging.....	40
2.1.5.4. Logistic Regression (LR)	41
2.1.5.4.1. Basic Framework.....	41

2.1.5.4.2. Combination of Instance-Based Learning and LR.....	42
2.1.5.5. Naïve Bayes Classifier (NBC).....	43
2.1.5.5.1. Advanced Description of NBC	43
2.1.5.5.2. Multinomial NBC	44
2.1.5.5.3. Naïve Bayes Classifier for Unbalanced Classes	44
2.1.5.6. K-Nearest Neighbor (KNN).....	46
2.1.5.6.1. Basic Concept of KNN.....	46
2.1.5.6.2. Weight Adjustment K-Nearest Neighbor Classification (WAKNN)	47
2.1.5.7. Support Vector Machine (SVM).....	48
2.1.5.7.1. String Kernel.....	49
2.1.5.7.2. Multi-Instance Learning (MIL)	49
2.1.5.8. Decision Tree	50
2.1.5.9. Random Tree	52
2.1.5.10. Deep Learning	53
2.1.5.10.1. Deep Neural Networks (DNN).....	53
2.1.5.10.2. Recurrent Neural Network (RNN)	54
2.1.5.10.3. Convolutional Neural Network (CNN).....	57
2.1.6. Training.....	58
2.1.6.1. Datasets.....	58
III. Summary of Techniques and Classifiers.....	62
1. Feature Extraction Comparison	62
2. Text Classification Comparison	63
IV. Methodology	64
1. Research Process.....	64
2. Research Methodology.....	64
3. Research Philosophy	65
4. Research Approach	66
5. Research Strategy	66
6. Data Collection and Data Analysis.....	66
7. Legal and Ethical Consideration.....	67
V. Implementation	67
1. Implementation Tools and Prerequisite Setup	67
1.1. Tools	67
1.2. Prerequisite Setup	67
2. Dataset Selection and Pre-Processing	68
2.1. Dataset Selection	68

2.2.	Pre-Processing, Indexing and Feature Extraction	69
3.	Classifier Identification and Training	71
4.	Design of the Chat Application	74
4.1.	Main Interface (Python & HTML).....	74
4.2.	Routes (HTML)	76
4.3.	Second Interface (HTML).....	77
4.4.	Events (Python)	78
4.5.	Blueprint (Python)	79
4.6.	Create the Application (Python)	79
4.7.	Chat Application (Python)	80
4.8.	Final Chat Application.....	80
4.9.	Demonstration and Evaluation of Offensive Language in Chat Application.....	83
VI.	Evaluation and Recommendation.....	85
VII.	Conclusion and Future Work.....	86
VIII.	References.....	87
IX.	Appendix A Project Log	95
X.	Appendix B Ethics Form	99

Table of Figures

Figure 1: NLP Techniques.....	14
Figure 2: Pattern-based Lemmatization Mapping Rule (Attia, et al., 2016).....	16
Figure 3: Example of a phrase with a word having two POS tags (Malhotra & Godayal, 2018).....	19
Figure 4: Part-Of-Speech tagging steps using HMM training and Viterbi algorithm (Pajarskaite, et al., 2004).	20
Figure 5: Parse Tree (Community, 2019).	22
Figure 6: Workflow of the proposed WSD framework (Wang, et al., 2020).	25
Figure 7: Subgraph construction for semantic path exploration (Wang, et al., 2020).....	27
Figure 8: NLG: A simple view (Bateman & Zock, 2012).	28
Figure 9: Document Classification Process (Mahender & Korde, 2012).	30
Figure 10: Generic Strategy for Text Classification (Dalal & Zaveri, 2011).	30
Figure 11: TC Process used for this research.....	31
Figure 12: Term-Document Matrix (Mahender & Korde, 2012).	32
Figure 13: Semantic Feature (Al-Makhadmeh & Tolba, 2020).	34
Figure 14: Text Classification methods, classification algorithms, techniques, and classifiers (Thangaraj & Sivakami, n.d.).	37
Figure 15: Boosting Technique Architecture (Kowsari, et al., 2019).	39
Figure 16: The AdaBoost Method Algorithm (Kowsari, et al., 2019).....	40
Figure 17: Simplified model of the bagging technique (Kowsari, et al., 2019).	40
Figure 18: Bagging Algorithm (Kowsari, et al., 2019).	41

Figure 19: Architecture of KNN for a 2D data set and Three classes (Kowsari, et al., 2019).....	47
Figure 20: Random Forest (Kowsari, et al., 2019).	52
Figure 21: Standard, fully connected Deep Neural Network (DNN) (Kowsari, et al., 2019).....	54
Figure 22: Standard LSTM/GRU Recurrent Neural Networks (RNN) (Kowsari, et al., 2019).....	55
Figure 23: LSTM cell (Kowsari, et al., 2017).....	56
Figure 24: GRU Cell (Kowsari, et al., 2017).....	57
Figure 25: Research process Flowchart.	64
Figure 26 Research Onion (Melnikovas, 2018).....	65
Figure 27: Requirements for Python code to function properly.	68
Figure 28: Install Packages in Requirements.txt	68
Figure 29: Custom vectorizer.....	70
Figure 30: Library Import.....	70
Figure 31: Custom Analyser for the vectorizer	71
Figure 32: SVM training used pre-processed dataset and scikit learn.....	72
Figure 33:Library Import 2	72
Figure 34: Vectorization of dataset and Training of Linear SVM.	73
Figure 35: LoginForm Python	75
Figure 36: Login Form HTML.....	75
Figure 37: Chat Application Login Interface.....	75
Figure 38: Routes.....	76
Figure 39:Chatting Interface.....	77
Figure 40: Main Interface (Visual Representation).	78
Figure 41: Code for a user joining the chatroom.	78
Figure 42: Code for a user leaving the chatroom.....	78
Figure 43: User joining and leaving a "cx" chatroom.....	79
Figure 44: Blueprint used to declare "Main"......	79
Figure 45: Code used to create the Application.	80
Figure 46: Chat Application Launcher.	80
Figure 47: User John logging into the chatroom.....	81
Figure 48: User Warren logging into the chatroom.....	81
Figure 49: Conversation between two users in the chat app.....	82
Figure 50: Embedding the pre-trained SVM model in the chat application.....	83
Figure 51: Default function used to send a message in the chatroom.....	83
Figure 52: Edited function to send a message in the chatroom.....	84
Figure 53: Chat with Text Classification Linear SVM embedded.....	84

Table of Tables

Table 1 Process of getting "Lemma" (Bitext, 2018).....	15
Table 2: Process of getting "Stem" (Bitext, 2018).....	17
Table 3: Part-Of-Speech Tag generated by NLTK package (Malhotra & Godayal, 2018).....	19
Table 4: Grammar table derived from the sentence.	21
Table 5: Analysis of some of the existing hate speech datasets (Madukwe, et al., 2020).....	58
Table 6: Hate speech detection Approach (Pamungkas, et al., 2020).....	60
Table 7: Feature Extraction Comparison (Kowsari, et al., 2019).....	62
Table 8: Text Classification Comparison.....	63

Acronyms

AI	Artificial intelligence
CNN	Convolutional Neural Networks
DNN	Deep Neural Networks
FDT	Fast Decision-Tree
FE	Feature Extraction
FIFO	First-In-First-Out
FS	Feature Selection
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
KB	Knowledge Bases
KNN	K-Nearest Neighbor
LPI	Locality Preserving Indexing
LR	Logistic Regression
LSHTC	Large Scale Hierarchical Text Classification
LSI	Latent Semantic Indexing
LSTM	Long Short-Term Memory
MIL	Multi-Instance learning
ML	Machine Learning
MLP	Multi-layer Perceptron
NBC	Naïve Bayes Classifier
NER	Named Entity Recognition
NLG	Natural Language Generation

NLP	Natural Language Processing
NLTK	Natural Language Toolkit
POS tagging	Part-Of-Speech Tagging
RBF Kernel	Radial Basis Function Kernel
RNN	Recurrent Neural Networks
RSTDT	Rhetorical Structure Theory Discourse Treebank
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TC	Text Classification
TF-IDF	Term Frequency-Inverse Document Frequency)
UPOS	Universal Part-of-Speech
VSM	Vector Space Model
WAKNN	Weight Adjustment K-Nearest Neighbor
WRV	Word Representation Vectors
WSD	Word Sense Disambiguation

I. Introduction

1. Background Overview/ Motivation

It is undeniable that social media improved the lives of millions of people around the world in terms of connectivity, education, promotion of self, promotion of businesses, ability to access information, awareness, promotion of social causes and many other aspects. Social media networks allow people to connect, discover and share their point of views and preferences irrespective of age and background. This freedom provided by social networks also implies that children and adolescent with basic knowledge of the internet can access social networks and be exposed to a lot. (Psychatry, 2018) indicates that 90% of teenagers between 13 and 17 have used social media, 70% report having at least one active social media profile, and 51% report visiting a social media site at least daily. The survey also identifies that on average, teens are online almost nine hours a day, not including time for homework. Furthermore, (Social, 2020) claims that 1 in 3 children are cyberbullied and 70% of children say have cyberbullied someone online. The statistical information mentioned previously indicates that teenage users are present in numbers on social media and are victims of cyberbullying and other types of abuse.

Although social media provide its users with an environment for growth and connectivity, it is critical to also provide a safer environment for the users of younger age in particular but also users of all age facing various social media-related problems such as flaming, trolling, harassment, cyberbullying, cyberstalking and cyberthreats. These expressions are simply used to refer to vulgar and offensive content. With that in mind, many researchers proposed methods to solve or at least reduce the problem of abuse faced by users on social media. Several researchers use advance methods such as sentiment analysis, data mining, content-based methods, graph-based methods, fusion-based methods, machine learning-based method and many more to solve the problem of abuse on social networks.

2. Aim

This dissertation aims at providing an efficient method to detect and prevent the use of abusive language over social media networks. Unlike, existing methods tackling the problem of social media abuse by focusing on comments and posts, the proposed method focuses on chat messages and proposes an efficient method to identify chat messages that are considered offensive. To achieve that a machine learning algorithm is trained and embedded in a chat application developed to demonstrate the way the algorithm functions.

Also, although a lot of research covers text classification, none have discussed the available Text Classification algorithms, classifiers, preprocessing techniques, and popular datasets used to improve on the text classification process in detail. In this research, this review is done and noticeable approaches to the text classification problem are also discussed.

3. Objectives

- To conduct a literature review on machine learning and AI in order to apprehend the way they operate and how they have been used in the context of text classification.
- Review available algorithms and dataset available to determine the appropriate one for machine learning and AI training purposes.
- To develop an approach (algorithm) to be able to detect and categorise offensive slangs and unacceptable language.
- To evaluate and test the accuracy of the algorithm with real-life scenarios.

4. Structure of the Dissertation

Following the **Declaration** and **Abstract** respectively, **Chapter 1** of the dissertation is the introduction. It first gives an **overview** of the **background** of the research and indicates the **motivation** behind the dissertation, followed by the **aim** of the research, **objectives**, and finally provides a detailed description of the **structure of the dissertation**.

In **Chapter 2**, a systematic literature review is done on the various Natural Language Processing (NLP) methods and the way those methods have been used in the context of the classification is discussed. Furthermore, a review of existing text classification processes is done, and a more suitable and detailed text classification process is

proposed. The stages of the proposed process are then described in detail and previous ways to complete them are reviewed. The most important part of chapter 2 is the review and analysis of available machine learning algorithms and datasets. A thorough analysis of the algorithms was first done indicating the way they function, their variations and the advantages and limitations they come with. Next, the commonly used datasets for the training of classifiers in the context of offensive language identification were reviewed. Finally, the proposed approach is presented into details and a demonstration of the way it functions is also provided.

Chapter 3 is a continuation of the discussion of the algorithms, datasets, and techniques of Chapter 2. In this section, the feature extraction methods are compared by presenting advantages and limitation they come with, followed by a comparison of text classification models discussed previously.

Chapter 4 gives an insight into the Methodology used for this dissertation and explains how it was determined. To have a strong and efficient Methodology, the **research process** to be used needs to be identified, followed by the selection of the appropriate **research methodology, research philosophy, research approach, research strategy** and model to be used for **data collection and analysis**. Finally, legal and ethical considerations are taken into consideration.

In **Chapter 5**, the text classification process, preprocessing and training techniques and classifiers proposed in **chapter 2** are used to prepare the dataset and train the algorithm. Furthermore, once the algorithm is trained and capable of identifying abusing language, the chat application used to simulate a real-life situation is built and the pre-trained model is embedded in the chat application making it capable of classifying offensive chat messages and the performance of the algorithm is evaluated in a real-life situation. In

Chapter 6, an evaluation of the implementation done is performed giving an insight on what was achieved during the implementation phase and what these achievements imply for the detection of abusive language using Text Classification. Furthermore, in this section some recommendations are proposed aiming at improving the work done.

Finally, in **Chapter 7**, an explanatory conclusion is done to summarize the work accomplishments done throughout the work and future work to be done are proposed to cover the gap the current research could not cover.

II. Literature Review

Over the past decades, technology impact on human's daily life has increased exponentially especially with the growth of social media giving users the ability to communicate, participate in social activities, share their opinions, feelings, and experiences. Social media essentially promoted freedom of speech, creative process and large-scale information sharing. However, the downside of this freedom was the ever-growing use of hate speech, abusive language, extremism, violence, fake news, and other similar problems. To alleviate this situation, various researchers turned to Natural Language Processing and more accurately to Text Classification for the detection of Abusive language, hate speech and others.

1. Natural Language Processing (NLP)

Natural Language Processing (NLP) is “a field within artificial intelligence (AI) and machine learning that combines linguistics and computer science to break down language, so it can be analysed by machines.” (MonkeyLearn, n.d.). To be specific, NLP aims at easing human and computer interaction by helping machines learn and understand human language. The goal is for machines to be able to “read, decipher, understand and make sense of the human languages in a manner that is valuable” (Dr. Garbade, 2018).

They are two main sets of techniques employed to complete NLP tasks: Syntactic Analysis and Semantic Analysis. Figure 1 gives an overview of the NLP most common techniques.

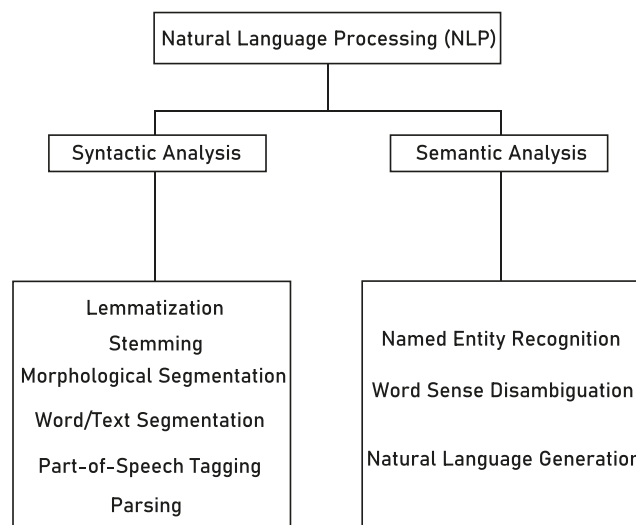


Figure 1: NLP Techniques.

1.1.Syntactic Analysis

Syntactic Analysis focuses on the syntax of sentences. Simply put, it focuses, on the way natural language and grammatical rules interact to make meaningful sentences. Through the use of algorithms, a set of words is associated with the appropriate grammatical rule and arranged for computers to process, understand, and assimilate for future use. Here are the main Syntactic Analysis techniques.

1.1.1. Lemmatization

Lemmatization refers to the process of identifying the base entry (lemma) from a word according to the dictionary meaning of the word (Suhartono, 2014). The lemma of a word contains its full meaning and grammatical part in a sentence (Dootio & Wagan, 2017). Considering the verb “to eat”, it has various forms based on the way it is used such as “eat, ate, eats and eating”.

In this context the lemma of the verb “to eat” is “eat”, its original and primary form. This process reduces the various forms of a word to facilitate analysis. Table 1 provides an overview of the way lemma is determined.

Table 1 Process of getting "Lemma" (Bitext, 2018).

Form	Morphological Information	Lemma
Eats	Third-person, singular number, present tense of the verb <i>eat</i>	Eat
Eating	Gerund of the verb <i>eat</i>	Eat
Studied	“Past” tense of the verb <i>study</i>	Study
Studies	Third-person, singular number, present tense of the verb <i>study</i>	Study

Although the Lemmatization process provides great advantages, it has a major limitation. It is effective for morphologically poor languages such as English but has a more difficult time with morphologically rich languages such as Arabic (Freihat, et al., 2018). To face this limitation, various researchers proposed **Tools** and **Approaches**. (Boudchiche, et al., 2017) proposes a tool called the **Alkhalil** lemmatizer. By applying a morpho-syntactic analysis to the text to generate a set of potentiation word surface forms and utilising the Hidden Markov Models to select a unique form from the set, the lemmatization process accuracy rose to 94%. Another proposed tool is **Madamira** (Pasha, et al., 2014). This tool performs two main tasks. Unlike the **Alkhalil** method, a simpler morphological analysis is first performed on the inputted sentence to determine a list of prospective analyses and then using language models, the word’s lemma is then identified.

This method improved the lemmatization process to 96.6% accuracy. In contrast, some tools use a different approach. By taking advantage of a dictionary of words and a discretization process functioning according to the number of occurrences per word. With this method, tools such as the **Farasa** lemmatizer managed to achieve a 97.32% accuracy (Freihat, et al., 2018).

Except for tools, some researchers propose certain approaches to improve on the Lemmatization process. (Attia, et al., 2016) proposes a pattern-based approach by using a machine learning classifier to predict the pattern of the lemma. Unlike in the **Alkhalil** lemmatizer (Boudchiche, et al., 2017), this process does not use lexicons and morphological analysis. Instead, it considers lemmatization as a classification problem and uses a two-level mapping process. On the first level, the stem is mapped to the pattern of the lemma. Then on the second level, the pattern of the lemma is mapped to the actual lemma's form by filling the slots in the pattern with radicals extracted from the stem. Figure 2 gives an overview of the two-level lemmatization employed in this method (Attia, et al., 2016).

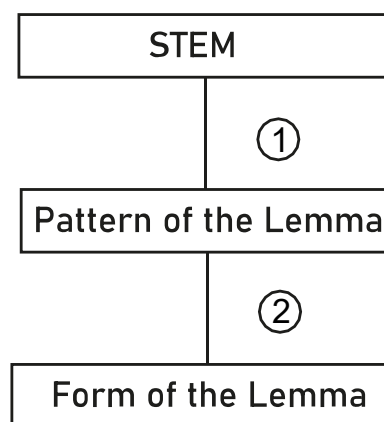


Figure 2: Pattern-based Lemmatization Mapping Rule (Attia, et al., 2016).

The lemmatizer used in (Al-Shammari & Lin, 2008) uses all the previously mentioned techniques such as word patterns, roots, syntactic and morphological basic rules to reduce words into their lemma canonical form. The difference with (Boudchiche, et al., 2017) and (Attia, et al., 2016) is that in this case, the lemmatizer tried to improve the extraction process for words that failed the rule-based analysis by using a dictionary approach just like in (Freihat, et al., 2018).

1.1.2. Stemming

Similar to lemmatization, stemming aims at reducing the number of variations of a word to a canonical representation (stem) (Suhartono, 2014). The stem is obtained by removing the last few characters of a given word based on a list of known prefixes and suffixes typical to the language being used (Bitext, 2018). It is done to obtain a root form of words, even if that form does not have any meaning. Table 2 gives an overview of the way stem is obtained.

Table 2: Process of getting "Stem" (Bitext, 2018).

Form	Suffix	Stem
Studies	-es	Studi
Studying	-ing	Study

Many researchers propose various methods to improve the results of the stemming process. In 1980, Porter introduces a stemmer called a "reducer stemmer". The stemmer is capable of identifying suffixes and ignore prefixes (Porter, 1980). Krovetz then introduces a method relying on a group of morphological rules and a dictionary approach to find the stem of words (Krovetz, 2000). Unlike Porter's method, Krovetz method can identify both word's prefixes and suffixes.

A similar approach to Porter's is the bottom-up approach introduced by (Sharifloo & Shamsfard, 2008). The method uses morphological rules to find stems of Persian words. Another well-known method is the co-occurrence based stemming algorithm proposed by (Gupta, et al., 2011). This method is capable of measuring how often a pair of words can occur in a document. To achieve that, an analysis is done on a certain corpus to measure the likelihood of the co-occurrence of two-word variants. The statistics are collected and used by the algorithm to predict the co-occurrence of two words in a text.

1.1.3. Morphological Segmentation

Morphological Segmentation is a technique that aims at breaking down words into individual units called morphemes (Dr. Garbade, 2018). Mainly used for information retrieval and text mining, morphological segmentation has two main advantages it allows algorithms to associate previously known forms of a word to their newly identified

variation and works as a mean to reduce data sparsity (Wang, et al., 2016). Mostly used by researchers for linguistic pre-processing, (Wang, et al., 2016) proposes the use of morphological segmentation in Novel Neural Network Architectures enabling them to learn the structure of inputted sequences from raw data and predict morphological boundaries. The Novel Neural Network relies on Long Short-Term Memory (LSTM) to accomplish morphological prediction and takes advantage of windows of characters to “capture more contextual information” (Wang, et al., 2016).

1.1.4. Word/Text Segmentation

Text segmentation is the process of extracting coherent blocks of text (P, et al., 2012). This method involves dividing a document into smaller parts referred to as “segments” or “segment boundary” (Pak & Teh, 2018). Widely used in text processing, this method splits big documents into segments with each segment having its appropriate meaning. Segments are then labelled either as words, sentences, topics, phrases, or any other label based on the requirements of the text analysis (Pak & Teh, 2018). This method presents two main advantages it first facilitates the understanding and processing of large blocks of documents through the use of segments and uses each segment as a unit of analysis improving access and processing capabilities. Text segmentation can be used for various applications. By using the three layers of information in natural language texts (syntactic, semantic, and pragmatic), (Wu, et al., 2007) uses text segmentation as a mean to perform emotion extraction.

To achieve that, (Wu, et al., 2007) uses text segmentation on the syntactic layer to process the text corpus. On the semantic layer, POS (Part-Of-Speech) tagging, and chunk analysis are used to label the linguistic features of the text. Finally, using the analysis from the semantic layers, words are labelled based on their emotional orientation on the pragmatic layer.

Information gathered from the various layers is then used to build a system capable of identifying the sentiment orientation of texts. This method takes advantage of all the layers and uses techniques specific to each of them to extract the best possible information from the inputted data.

Other research use text segmentation for sentiment mining (Gao, et al., 2010) (Xia, et al., 2010), opinion mining (Liu, et al., 2006) (Osman & Yearwood, 2007), topic identification (Brants, et al., 2002) (Flejter, et al., 2007) , language detection (Potrus, et al., 2014) and information retrieval (Huang, et al., 2003).

1.1.5. Part-of-Speech Tagging (POS)

POS “is a process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context – i.e., its relationship with adjacent and related words in a phrase, sentence, or paragraph” (Malhotra & Godayal, 2018). Simply put, it is the process of identifying and classifying words as either noun, verbs, adjectives, adverbs, etc. The difficulty with this method is the complexity of languages. A word can have different meanings based on the context in which it is used resulting in a single word having multiple part-of-speech tags. To resolve this ambiguity, this process relies on machine-based part-of-speech tagging.

Figure 3 gives an overview of a sentence where a single word is used twice and has a different meaning each time.

They refuse to permit us to obtain the refuse permit.

Figure 3: Example of a phrase with a word having two POS tags (Malhotra & Godayal, 2018).

In Figure 3, the first “refuse” is used as a verb to mean “to deny”, however, the second “refuse” is used are a noun to mean “trash”. This context is a typical example of why POS Tagging is required to differentiate each word, their meaning and type.

Table 3: Part-Of-Speech Tag generated by NLTK package (Malhotra & Godayal, 2018).

Inputted Text	Part-of-Speech Tag Using with NLTK Package
They refuse to permit us to obtain the refuse permits.	>>> text = word_tokenize("They refuse to permit us to obtain the refuse permit")>>> nltk.pos_tag(text) [('They', 'PRP'), ('refuse', 'VBP'), ('to', 'TO'), ('permit', 'VB'), ('us', 'PRP'), ('to', 'TO'), ('obtain', 'VB'), ('the', 'DT'), ('refuse', 'NN'), ('permit', 'NN')]

Table 3 gives an overview of the part-of-speech tag process applied to a sentence using the NLTK (Natural Language Toolkit) package. The result shows that for each “refuse”, a different tag was assigned. For the first “refuse”, the “VBP” tag was assigned, standing for “Present Tense Verb”. For the second “refuse” the “NN” tag was assigned, standing for “Noun”. Although both words are the same the context makes a difference and allows to make a clear differentiation for future use. To accomplish Part-of-Speech Tagging, two main steps are needed: morphological analysis and disambiguation. “A word is considered ambiguous if it can be assigned to more than one morphological category” (Pajarskaite, et al., 2004). The most common type of morphological ambiguities are Homofoms, Homographs, Gender ambiguities and Case Ambiguities.

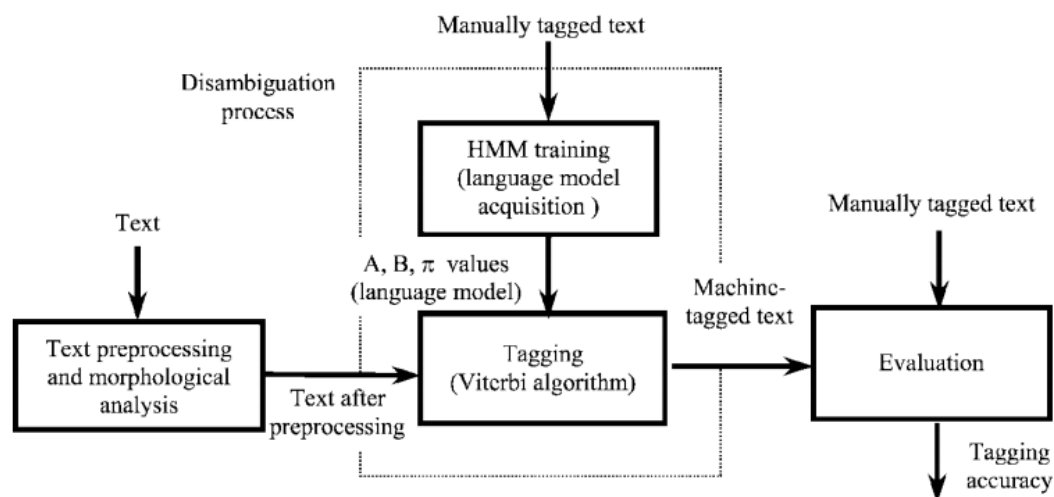


Figure 4: Part-Of-Speech tagging steps using HMM training and Viterbi algorithm (Pajarskaite, et al., 2004).

(Malhotra & Godayal, 2018) claims that Part-of-Speech Tagging algorithms fall into two groups: Rule-based POS Taggers and Stochastic POS taggers. To achieve a higher success rate during the disambiguation process, (Pajarskaite, et al., 2004) proposes a process of stochastic disambiguation using the Hidden Markov Model (HMM). Figure 4 shows the three main stages in the proposed POS tagging method: The Text Pre-processing and Morphological Analysis stage, the Disambiguation Process stage, and the Evaluation stage.

During the Text Pre-processing stage, words, and tokens relevant to the text are identified and sentence boundaries are marked. Following the previous stage, the morphological analysis is done, and annotations are assigned to each word or token previously identified.

To perform the morphological analysis, (Pajarskaite, et al., 2004) uses a tool called “Lemuoklis” in charge of assigning all tags and other morphological information related to each word/token.

Next is the Disambiguation Process. There are two basic phases: the training and tagging phase. The probability of tag-tag and tag-word pairs occurring is determined during the training process and stored in matrices to form a language model. The language model is then applied during the tagging phase to determine the best possible tag to be assigned to each word/token in the document.

Three experiments are then performed during the Evaluation stage to determine the efficiency of the method: “the changing language models (bigram and trigram), changing smoothing methods (Add-One and Good-Turning) and changing the size of training corpus (50%, 75%, 100% of the original training corpus)” (Pajarskaite, et al., 2004).

1.1.6. Parsing

Parsing “is the automatic analysis of a sentence with respect to its syntactic structure” (Universitet, n.d.). In short, it implies determining a text's syntactic structure by performing a grammatical analysis of its words. The result of the analysis and the parsing process is displayed in a parse tree. The tree then displays the grammar rules applied to each part of the sentence. For the following sentence: “Tom ate an apple”, Table 4 displays the grammar table determined from the analysis. Each row indicates grammar rules applicable to each word or group of words of the sentence.

Table 4: Grammar table derived from the sentence.

Sentence	noun_phrase, verb phrase
Noun_phrase	proper_noun
Noun_phrase	determiner, noun
verb phrase	verb, noun_phrase
Proper_noun	[Tom]
noun	[apple]
verb	[ate]
determiner	[an]

The grammar table introduces three elements: “sentence” as the **root** of the parse tree, “noun_phrase” and “verb_phrase” have children so are **non-terminals** and “Tom”, “ate”, “an” and “apple” are **terminals**. These are used to form the parse tree displayed in Figure 5.

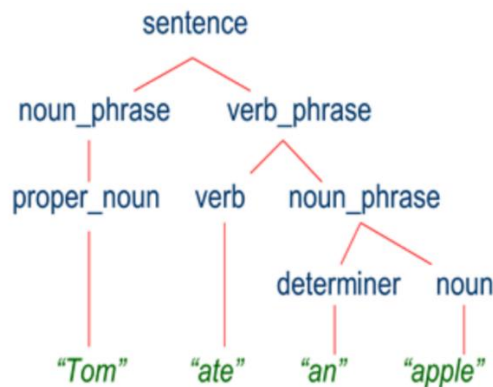


Figure 5: Parse Tree (Community, 2019).

Due to the complexity of words and sentences, parsing is not always an easy process. Many researchers propose advance methods to successfully achieve parsing. (Dootio & Wagan, 2019) proposes a hierarchical syntactic parsing method based on FIFO (First-In-First-Out) data structure. In this method, all tokens depend on each other and following the FIFO logic, every time the first word is processed, it is mapped with “**phrase**” and “**Universal Part of Speech (UPOS)**” tag. In the case of unknown tokens, tokens are marked with an “X” by the parser to indicate a UPOS tag. After the parsing is completed, the results are compiled and used to construct multi-class and multi-feature datasets. With the datasets established, supervised machine learning techniques (SVM, Random Forest and K-NN) are used and trained to perform a critical analysis and a thorough evaluation of the datasets. SVM was reported to deliver better performances on class phrases as a result of the implementation of the hierarchical parsing process, while Random Forest also achieved an excellent performance but on class TagStatus.

Another method of advanced parsing is the one proposed by (Hou & Lu, 2020), a rhetorical parsing. The rhetorical parsing aims to understand “how two text spans are related to each other in the context” and improve NLP applications by performing in-depth analysis and identifying meaningful information in large documents. (Hou & Lu, 2020) covers the gap in the existing research by proposing a method applied to the translation of the Chinese language since almost all the existing parsing method is for the English language.

For the rhetorical parsing to function efficiently, it first adopts the RST-DT framework and embeds Chinese and English text in the same latent space allowing the rhetorical relations between the Chinese text spans to be determined by the rhetorical relations in the RST-DT framework. The experimental results of this method resulted in performances of the same level as existing methods.

1.2.Semantic Analysis

“In general linguistics, semantic analysis refers to analysing the meanings of words, fixed expressions, whole sentences, and utterances in context.” (Goddard & Schalley, 2010). In other words, semantics analysis focuses on the meaning of the text. (Dr. Garbade, 2018) claims that semantics analysis is “one of the most difficult aspects of Natural Language Processing (NLP) that has not been fully resolved yet”. Here are the main Semantics Analysis techniques.

1.2.1. Named Entity Recognition (NER)

The Named Entity Recognition technique is one of the most advantageous technique when it comes to semantic analysis. Its main focus is to identify relevant names in a text and classify them based on the entity they belong to (Jain, et al., 2018). NER model work in two phases.

During the first phase, the algorithm divides the text into small segments and classifies them based on predefined categories (name, occupation, location etc..) regardless of their format (bold, capitalised etc..). The second phase depends on the task the algorithm is used for. NER can be used for language and speech processing with the use of a user preference graph to improve smart replies of search suggestions (Jain, et al., 2018) and many more.

NER algorithms can be used in various cases. Here are a few examples and usage of the NER algorithm applications.

- **News Categorization:** Used for news categorization, NER algorithm performs an automatic scan of news articles and identifies critical information based on their category such as individuals name, companies, organisations, people, celebrities name, places etc. This allows to classify news articles based on their content and provide users with articles specific to their preferences (Kumawat, 2019).

- **Efficient Search Engine:** NER algorithms boost the efficiency of search engines by applying a methodical analysis, classifying, and storing articles, news and search results into specific tags (Kumawat, 2019). This improves the response time of search engines and overall performances.
- **Customer Support:** By implementing NER API in social networks where heavy/slow traffic is a problem, traffic engineers can be quickly informed of heavy traffic with the algorithm pulling keywords and assign computational resources accordingly and consequently improve customer experience and service.

1.2.2. Word Sense Disambiguation (WSD)

Previously discussed as “disambiguation process”, Word Sense Disambiguation can be defined as “the ability to identify the meaning of words in context in a computational manner” (Navigli, 2009). The goal of NLP is for machines to be able to understand and make sense of human languages. However human languages are ambiguous and not always easy to process. A perfect example is the following: (a) I can hear **bass** sounds and (b) They like grilled **bass** (Navigli, 2009). Both words are written the same way but have a different meaning due to the context they are used into. The first **bass** refers to a certain sound type while the second one refers to a type of fish. This ambiguity of human language represents a huge obstacle to achieve NLP’s goal.

WSD has proven to reduce the problem that represents language ambiguity. It is a method highly used for “Sentiment Analysis” (Chihli Hung, 2016), “Information Retrieval” (Zhong & Ng, 2012), “Information Extraction” (Bovi, et al., 2015), “Machine Translation” (Xiong & Zhang, 2014), “Knowledge Graph Construction” (Raganato, et al., 2016) etc.

Most of the WSD solutions proposed, follow a supervised approach or a knowledge-based approach.

- For the supervised approach, WSD is addressed as “a classification problem with each classifier focusing on one word”. The classifiers are separately trained beforehand with annotated samples and designed to address a particular word (Wang, et al., 2020).

- The goal of the knowledge-based approach is to devise methodologies capable of making full use of the Knowledge Bases (KB) such as WordNet and BabelNet. It is important to mention that KB's are "lexical database dictionaries for the English language designed for natural language processing". For the knowledge-based approach, there are two practice trends. The first one considers the overlap between the context of an ambiguous word and its potential meaning and sub-meanings derived from KB databases. The most similar meaning is then selected and considered as the predicted and definitive meaning. The second trend uses a graph-based model. A graph is drawn based on the word's context and other connections retrieved from the KB database. Graph-based algorithms such as PageRank (Brin & Page, 1998) and Latent Dirichlet Allocation (Blei, et al., 2003) are then used to predict the meaning of other words using the pre-built graph as a model.

Despite the advantages of these two approaches, the primary challenge is to design a system capable of taking full advantage of the information and knowledge KB's have to offer and to enable machines to perform disambiguation at a human's level.

(Wang, et al., 2020) proposes a framework based on the all-word WSD principle. The framework has four main components: "Semantic Space Exploration, Relation Extraction, Similarity Calculation and Semantic Path Exploration". Figure 6 gives an overview of the workflow of the WSD framework.

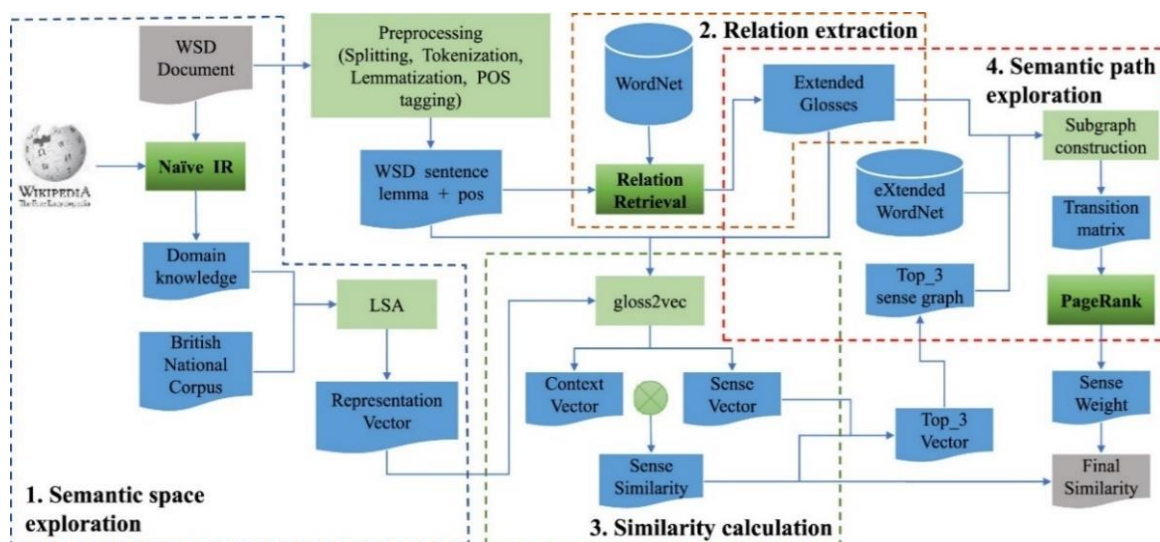


Figure 6: Workflow of the proposed WSD framework (Wang, et al., 2020).

There are four components to the WSD framework:

- **Semantic Space Exploration** works mainly with WSD documents. By applying a Naïve IR, domain knowledge documents are extracted from large documents corpus and then combined with KB's such as the British National Corpus using Latent Semantic Analysis. This is done to identify and learn Word Representation Vectors (WRV) in documents. (Basile, et al., 2014) and (Camacho-Collados, et al., 2016) demonstrate that WRV's are critical elements, used to improve word disambiguation. The vectors will be used in the following components.
- **Relation Extraction** works with lemma, POS and a Synset Retrieval Algorithm. Once all WSD documents are processed, lemma and POS are extracted and grouped by sentences. The lemma and POS are then used to extract potential senses for each ambiguous word from KB's such as WordNet. Furthermore, each sense is assigned an extended gloss that can be obtained through the use of a synset retrieval algorithm. The gloss functions as a "bag of words" containing original or extended senses, an example of usage and variations of words contained in the pre-processed WSD documents.
- **Similarity Calculation** works with unified vectors, word vectors and dot product. During the previous stages, the following word vectors were identified: the word collections from the context and the extended gloss of each ambiguous word. This stage first aims at matching the word vectors with their respective unified vectors by applying a weighting strategy to the word vectors. After both vectors have been identified, the next step is to determine the dot product. The dot product is the similarity value determined from both vectors that indicate that a sense is the appropriate one. It is needed to determine the rank of potential senses for each word.
- **Semantic Path Exploration** is the most critical part of the framework. It works with extended gloss, sentence-level network, synsets, PageRank and WordNet KB. This stage is about capturing semantic paths in sentences. However, this process is complex and requires several steps. For the first step, extended gloss from the *Relation Extraction level* is extracted and embedded into a sentence level network

for additional disambiguation. The network can be built using the top 3 senses and related synsets derived from the sentences this choice is made to optimize efficiency.

During the second step, two elements come into play: WordNet and PageRank. WordNet impacts the gloss and synsets in two ways. First, WordNet turns the gloss and related synsets into “disambiguated synsets”. It then provides a connection between the newly established synsets and ensures that the connection is stable. On the other hand, PageRank determines and decides the appropriate meaning of the senses within the network.

Finally, the node (sense) within the network is used as a template to adjust previous similarities. Figure 7 gives an overview of the *Semantic Path Exploration* component of the framework.

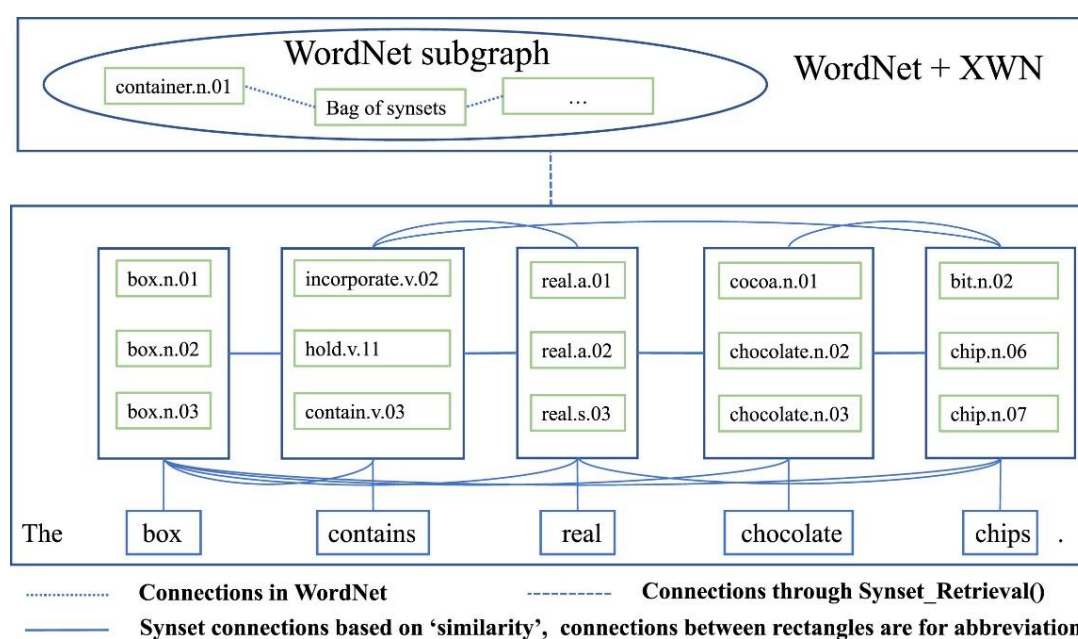


Figure 7: Subgraph construction for semantic path exploration (Wang, et al, 2020).

1.2.3. Natural Language Generation (NLG)

Natural Language Generation (NLG) is “a technique that converts raw structured data into plain English (or any other) language” (Kumawat, 2019). Also called “Storytelling”, NLG’s main focus is to make data more understandable. To that end, it processes and generates natural language based on four levels: “syntax level (structure of words), semantics level (the meaning of groups of words), pragmatics level (the intent of the groups of words) and dialogue level (the exchange of groups of words between people)” (Woolf, 2009).

Due to the complexity of languages, the task of NLG faces a lot of constraints and NLG needs to cover the language spectrum between the moment an action is being planned to the moment it is executed. In other words, the data needs to be mapped from its non-linguistic form (raw data from KB's) to its related linguistic form (Bateman & Zock, 2012). To achieve that, NLG performs a process of decision making and information processing. The process starts by determining and structuring the text's content, followed by a rhetorical structuring of the content at various levels (text, paragraph, sentence) (Bateman & Zock, 2012). Once the text's structure is decided and the text structured, the selection stage is initiated (Bateman & Zock, 2012). During which, "words and syntactic structures (word order, morphology, and grammatical constructions) are identified and finally the text layout (title, headers, footnotes, etc.) and acoustic patterns (prosody, pitch, intonation contour). are determined" (Bateman & Zock, 2012). Figure 8 gives a simple view of a standard NLG process.

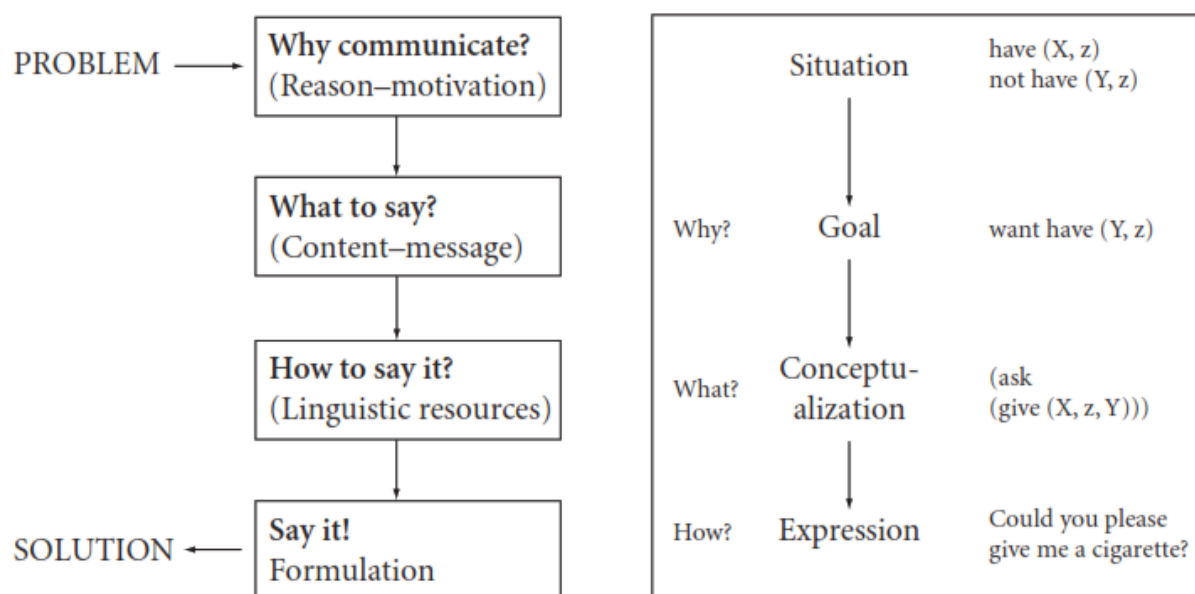


Figure 8: NLG: A simple view (Bateman & Zock, 2012).

Although NLG provides various advantages such as its ability to convert structure data into natural languages to improve language pattern understanding (Kumawat, 2019), it faces significant challenges. The most substantial challenge faced by NLG is the difficulty to provide an environment with a strong and flexible architecture for the various decisions made during the process to coexist while generating human-like languages within an adequate time frame.

The second challenge is the selection of the decisions involved in NLG. Due to NLG's task, there is various area of expertise that are required such as "knowledge of the language (lexicon, grammar, semantics), knowledge of the domain (what to say, relevance), strategic rhetorical knowledge (how to achieve communicative goals, text types style), engineering knowledge (how to decompose, represent and orchestrate the processing of information) and knowledge about the habits and constraints of end-user as an information processor (sociolinguistic and psychological factors)" (Bateman & Zock, 2012). All these variations, knowledge and area of expertise imply that the decisions selected need to be accurate and most relevant for the NLG to function.

With the aforementioned techniques, NLP combined with AI and machine learning has various applications in our daily life. The most common applications are "Sentiment Analysis, Chatbots and Virtual Assistants, Text Classification, Text Extraction, Machine Translation, Text Summarization, Market Intelligence, Auto-Correct, Intent Classification, Urgency Detection, Speech Recognition" (Wolff, 2020). This research work focuses on Text Classification applied for the classification and detection of offensive slangs and unacceptable language.

2. Text Classification

Text Classification is "the process of automatically assigning predefined tags or groupings to the text that relates to its content" (Wolff, 2020). It is an important application of NLP and can be implemented to complete various tasks such as "email classification (spam and non-spam), sentiment detection, blog post-classification, automatic tagging of customer queries etc." (Malik, n.d.). This section reviews the Text Classification (TC) process and discusses the various approaches attempted by researchers to improve TC results.

2.1.Text Classification Process

(Mahender & Korde, 2012) states that the TC process is made of six main stages as displayed in Figure 9.

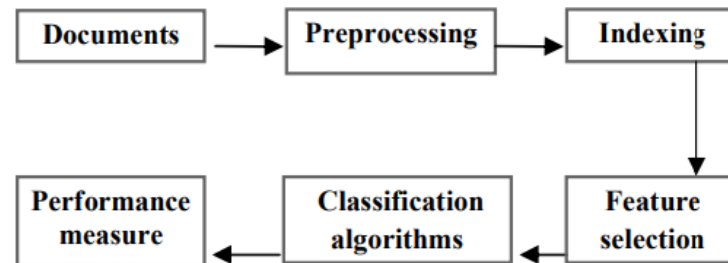


Figure 9: Document Classification Process (Mahender & Korde, 2012).

However, (Dalal & Zaveri, 2011) proposes a more generic and detailed process illustrated in Figure 10. The two stages have similarities but Figure 10 stages directly refer to the tasks performed at each stage while Figure 9 gives a more detailed process but fails to clarify the content of those steps from a diagram perspective.

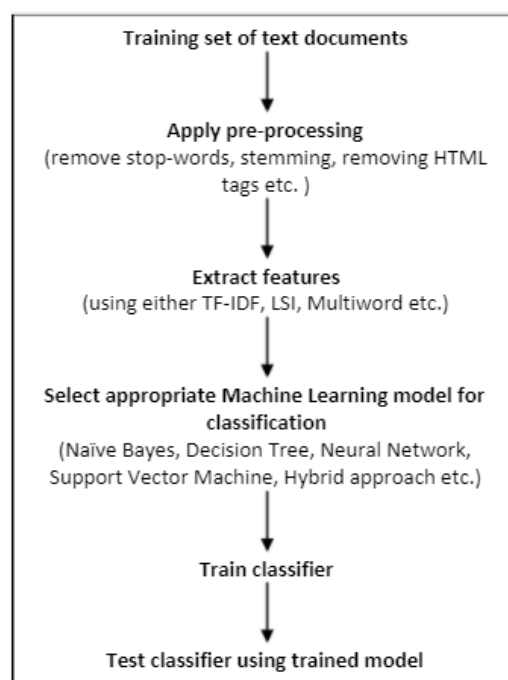


Figure 10: Generic Strategy for Text Classification (Dalal & Zaveri, 2011).

From these two stages, the following process was proposed as an alternate containing both generic and recent stages. Figure 11 gives an overview of the stage of TC discussed in this section.

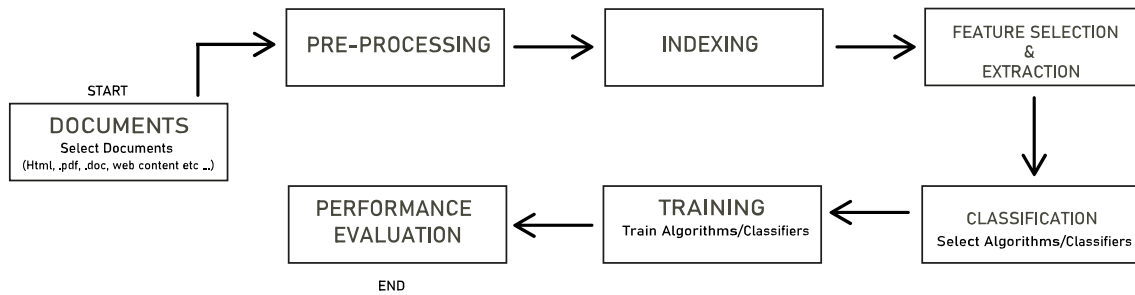


Figure 11: TC Process used for this research.

2.1.1. Document Collection

The focus of this stage is the collection of documents irrespective of their formats (HTML, .pdf, .doc, web content etc..). The documents are used to train the algorithm into understanding natural language. Once the documents to be used are identified, they go through the pre-processing process. The pre-processing process has two main stages the **Pre-Processing** and **Indexing** stages.

2.1.2. Pre-Processing

This stage aims at processing the text in the selected documents into a less ambiguous format. The pre-processing stage reduces the complexity of the text and makes it easier to handle by the algorithm. The most common methods used in pre-processing are **Tokenization** and **Stemming Word** along with all the previously discussed NLP techniques (**Semantic** and **Syntactic Analysis** techniques).

2.1.3. Indexing

This is the third stage of the TC process. Just like the previous stage, the aim is to reduce the complexity of the text and facilitate its usage by the algorithm. To that end, the indexing stage focuses on changing the documents into documents vectors by relying on **Document Representation**.

A popular document representation method proposed by (Mahender & Korde, 2012) is the Vector Space Model (VSM). “Vector-Space Model (VSM) for Information Retrieval represents documents and queries as vectors of weights. Each weight is a measure of the importance of an index term in a document or a query, respectively” (Melucci, 2009). In other words, to prevent losing the semantic relationship between the terms, VSM assigns weights to each term in the text. For the appropriate weight to be assigned

to the terms, the documents are represented into the vector space model following a Term-Document Matrix model illustrated in Figure 12

$$\begin{pmatrix} T_1 & T_2 & \dots & T_{at} & C_i \\ D_1 & w_{11} & w_{21} & \dots & w_{i1} c_1 \\ D_2 & w_{12} & w_{22} & \dots & w_{i2} c_2 \\ \vdots & \vdots & \vdots & & \vdots \\ D_n & w_{1n} & w_{2n} & \dots & w_{in} C_n \end{pmatrix}$$

Figure 12: Term-Document Matrix (Mahender & Korde, 2012).

In the method proposed by (Mahender & Korde, 2012), the vector space model, documents and queries are both vectors and each term (i) and query (j) is given a weight (W_{ij}). The weight is determined and assigned based on the frequency of the term (i) in the document as well as the query (j) and the collection. To determine the weight many techniques can be used but the most common ones are the Boolean weighting, word frequency weighting, tf-idf, entropy etc. Although VSM has proven to be an efficient process for indexing, it has a major drawback. Due to the use of a matrix-based model, VSM “results in a high sparse matrix which raises a problem of high dimensionality” (Mahender & Korde, 2012). To achieve an effective Indexing process, (Harish, et al., 2010) proposes various alternate methods to VSM:

- The first method focuses on making sure the semantic relationship between the terms in the document is maintained by using an ontology representation for the document (Harish, et al., 2010).
- The second method identifies a sequence of symbols known as N-Grams extracted from a string of words in a document (Harish, et al., 2010).
- In the third method, an automated text extraction algorithm is used to extract terms directly from the documents. It also defines word terms as vector components just like the VSM (Harish, et al., 2010).
- The fourth method uses Latent Semantic Indexing (LSI). LSI is “an indexing and retrieval method that uses a mathematical technique called Singular Value Decomposition (SVD) to identify patterns in the relationships between the terms and concepts contained in an unstructured collection of text” (Deerwester, 1988). To be more specific, text documents are first analysed to extract the context of the

documents. The documents are then grouped and categorised on the concept and the scores generated by the factor.

- The fifth method is the Locality Preserving Indexing (LPI). It is used in addition to LSI to efficiently determine the semantic structure of text documents (Mahender & Korde, 2012).

Although the methods have various advantages, the drawbacks are also critical. As previously mentioned, the main drawback with VSM is the high sparse matrix. For the second method, the main drawback is the difficulty to determine the right number of N-Grams to perform an effective document representation. The fourth method's drawback is its inability to discard the documents main features since its process identifies and preserves the main features of the documents for further processing. The fifth method is used as a solution for the LSI methods drawback. However, the LPI method also has a fatal drawback. It is not efficient in time and memory.

2.1.4. Feature Extraction/Selection (FE/FS)

Feature Extraction is one of the most important stages of text classification. The aim is to create a structured set of features for the training of the text classifier. To that end, the most relevant features to the classification process need to be identified and clustered into sets. FE can be summarised in two main phases. In the first phase, a measurement and scoring system is designed and calibrated based on the importance of certain words. In the second phase, the predetermined system is applied to the text document and scores are given to words in the text. Words with the highest score are then kept for further processing. This is done to reduce the high dimensionality of text features problem and remove irrelevant features.

There are various advanced FE methods such as the Gini Index, the Information Gain, the Mutual Information, the X^2 -Statistic, the Feature Transformation Methods: Supervised LSI, the Supervised Clustering for Dimensionality Reduction, the Linear Discriminant Analysis, and the Generalized Singular Value Decomposition. Many other approaches are proposed and adopted by researchers to improve and personalize the Feature Extraction process to their needs.

(Al-Makhadmeh & Tolba, 2020) proposes a method of automatic detection of hate speech in Twitter data. It argues that by applying NLP techniques to twitter-data, four

different features were identified: semantic feature, sentiment-based feature, unigram, and pattern feature. These features were identified by emphasizing the NLP techniques on particular expressions and the emotional level associated with them. Once identified, the features can be used to predict hate content.

The semantic features perform a simple task: identify interjections, capital words, punctuations etc. Figure 13 illustrates the result of a semantic feature applied to a text (Twitter tip Hate it).

No	Semantic Feature List	Example (Twitter tip Hate it)
1	Total words in the tweet	4
2	Number of question marks in a tweet	Nil
3	Total number of commas in a tweet	Nil
4	Number of exclamations in a tweet	Nil
5	Full stop in a tweet	1
6	Capital words in a tweet	1-Hate
7	Total interjection in a tweet	1-Hate
8	Total number of expressions in a tweet	1-Hate

Figure 13: Semantic Feature (Al-Makhadmeh & Tolba, 2020).

After the analysis of the semantic features, the sentiment features are analysed. This task is critical because sentiment features reflect the emotional level of the processed tweets. To that end, tweets are first categorised as either negative, positive, or neutral. They are then assigned an estimate score based on pre-processed words classified as positive words (P) and negative words (N). The score reflects the sentiment conveyed by the tweet. The score is given using the following equation.

$$pn(t) = \frac{P - N}{P + N} \quad (1)$$

Although equation (1) is the main method used to assign scores to tweets, various sentiment features such as the quantity of positive and negative words, emoticons and hashtags highly impact the polarity of a tweet. In the methods proposed by (Al-Makhadmeh & Tolba, 2020), a negative value between -5 to -1 is assigned to negative words, slang, and emoticons, while a positive value of 1 to 5 is assigned to positive words, slang, and emoticons. Following that process, sentiment features ensure the efficient and accurate prediction and identification of hate speech in social media. Just like the semantic features, the unigram features are derived from the tweets.

However, unlike in the semantic features process, the extraction of unigram features is done using POS tags and tweets are assigned a false and true value making it possible to identify negative and positive comments in the sentences. Unigram features aim to evaluate the probability of a word appearing in a tweet. To that end, Unigram features use a method where each word (w) is classified in a class C_i with its value being either ***neither***, ***offensive*** and ***hate language***. The process enables the model to determine the probability (ratio) of a word occurring in each tweet. With the probability identified, N_i refers to the numbers of words in the class i .

$$p_{12}(w) = \frac{N_1(w)}{N_2(w)} \quad (2)$$

$$p_{13}(w) = \frac{N_1(w)}{N_3(w)} \quad (3)$$

In equation (2), the “denominator ratio is initiated at 0 and then set at 2” (Al-Makhadmeh & Tolba, 2020).

The process is then repeated for each word until the condition in equation (4) is satisfied.

$$p_{ij}(w) \geq Th_u \quad (4)$$

In equation (4). p_{ij} is set for the relationship between two words in a class while Th_u can be defined as “the ratio of class values” (Al-Makhadmeh & Tolba, 2020) and is set by default to 1.4.

The last feature is the pattern feature. Derived from the training set using sentimental and non-sentimental features, pattern features are identified by processing each word in a tweet using POS tags. Each word is then marked by tags from two different sets. In the first set are simple classes such as noun, adverb, adjective and verb. The second set, however, introduces sentiment analysis by assigning words with the sentimental and non-sentimental class. One more aspect is considered in this method: word length. The maximum word length in this process is 7.

In the occurrence of a word with a length above 7, the word is either discarded from the list or other pattern features such as the one presented in (Razavi, et al., 2010) are derived and applied. The ratio, in this case, is identified as follows:

$$p12(pattern) = \frac{N1(pattern)}{N2(pattern)} \quad (5)$$

$$p13(pattern) = \frac{N1(pattern)}{N3(pattern)} \quad (6)$$

Just like in the unigram features equation, the “denominator ratio is initiated at 0 and then set at 2” (Al-Makhadmeh & Tolba, 2020). The process is then repeated for each word until the condition in equation (7) is satisfied.

$$p_{ij}(pattern) \geq Th_{pattern} \quad (7)$$

In equation (7). p_{ij} is set for the relationship between two patterns in a class while $Th_{pattern}$ can be defined as “the ratio of class values” (Al-Makhadmeh & Tolba, 2020) and is set by default to 1.4.

After applying the features to 1538 words, 1538 uniform features and 1873 patterns were identified. For each tweet containing a pattern, the value 1 was assigned while for tweets without one 0 was assigned. Also, the total number of tweets is n out of N and its maximum and default value is represented by $\alpha * \frac{n}{N}$ ($\alpha = 0.1$).

Although the semantic, sentiment, unigram and pattern features demonstrated great results, the process can be further optimised by using machine learning applications.

2.1.5. Classification

This stage can be defined as the process of classifying documents into predefined categories. The stage is the most important pillar of the TC. There are two main approaches to **Statistical** and **Machine Learning** (ML). In the Statistical approach, all tasks are done manually. Unlike in the Statistical Approach, in the Machine Learning approach, tasks are done automatically. (Allahyari, et al., 2017) argues that ML techniques were specially designed for automation. All tasks are automated, improving accuracy and reducing time consumption.

Moreover, the Machine Learning process is most used by researchers and (Thangaraj & Sivakami, n.d.) identifies three main types of machine learning algorithms: unsupervised, semi-supervised and supervised.

Figure 14 gives an overview of the most common methods, techniques, classification algorithms and classifiers used for Text Classification (TC). This section starts with a detailed description of the Rocchio algorithm followed by a description of two ensemble learning algorithms (Boosting and Bagging). After the ensemble learning algorithms, the following methods are reviewed: Logistic Regression, Naïve Bayes and K-Nearest neighbour. The Support Vector Machines (SVM) algorithm, a traditional yet widely used by the scientific community is also analysed as well as Tree-based Classification Algorithms such as the Decision Tree and Random Forests (known to be fast and reliable).

Finally, Deep Learning models/algorithms are also reviewed in particular the “Deep Neural Networks (DNN), Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN)”.

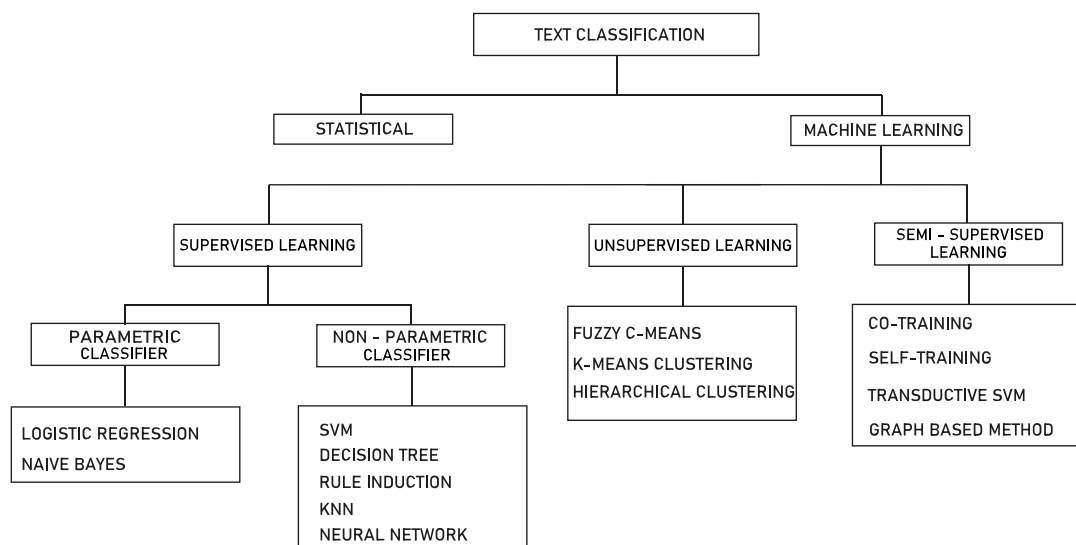


Figure 14: Text Classification methods, classification algorithms, techniques, and classifiers (Thangaraj & Sivakami, n.d.).

2.1.5.1. Rocchio Algorithm

Rocchio’s learning algorithm was first introduced in 1970 by J.J Rocchio (Rocchio, 1971) as a mean to use “relevance feedback in querying full-text databases” (Rocchio, 1971).

Mainly used for document routing in information retrieval, this method unlike other grouping techniques uses TF-IDF weights for each word instead of standard Boolean features. The Rocchio algorithm function following two stages.

During the first stage, training sets of documents are used to build a prototype vector for each class. In the second stage, test documents are assigned to classes based on the highest level of similarity between documents and the prototype vectors. The centroid of a class c is then determined by the average vector using the following equation:

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}_d \quad (8)$$

In the equation, " D_c stands for the set of documents D belonging to the class c and \vec{v}_d stands for the weighted vector representation of the document d (Kowsari, et al., 2019). As for d , it represents one the smallest distance between a document and the centroid c^* :

$$c^* = \arg \min_c || \vec{\mu}_c - \vec{v}_d || \quad (9)$$

Centroids can be normalized as follows:

$$\vec{\mu}_c = \frac{\sum_{d \in D_c} \vec{v}_d}{|| \sum_{d \in D_c} \vec{v}_d ||} \quad (10)$$

Finally, the label of test documents can be identified following equation:

$$c_* = \arg \min_c \vec{\mu}_c * \vec{v}_d \quad (11)$$

Since Rocchio's publication in 1971, many researchers such as (Partalas, et al., 2015) and (Sowmya & Srinivasa, 2016) proposed various methods and processes to improve the results of Rocchio's algorithm. (Partalas, et al., 2015) proposes a method that focuses on datasets used by the algorithm.

In the research paper, a detailed description of the process used to build the hierarchical datasets is given. For the evaluation stage, two evaluation measures are used: the flat and hierarchical. For the flat evaluation method, predictions are correct if labels are included in the gold set and wrong if not included. On the other hand, in hierarchical method considers the hierarchical relations between the predicted labels and the gold ones. Resulting in predictions that can either be right, wrong or partially wrong.

After submitting this method to the LSHTC challenge, the flat and hierarchical method performed equally in the LSHTC1 challenge. However, in the LSHTC2 the flat method performed better. Finally, in the LSHTC3, the hierarchical method performed the best. Although the Rocchio algorithm provides various advantages it also faces several limitations.

Limitations

Rocchio algorithm has many limitations but the most noticeable ones are first the limited number of retrievable documents especially in the Rocchio algorithm model proposed by (Selvi, et al., 2017). Additionally, on this algorithm, only a semantic approach can be taken to get viable results (Albitar, et al., 2012).

2.1.5.2. Boosting

Boosting and Bagging can be classified as “voting classification techniques” “developed for document and text data set classification” (Farzi & Bolandi, 2016). The main difference between these two techniques is that boosting improves the distribution of training sets by adapting its process based on the performance of previous classifiers while bagging does not consider the other classifiers.

First introduced as a technique for boosting the performance of a weak learning algorithm”, the boosting algorithm was further improved by researchers such as Freund (Freund, 1992) and (Bloehdorn & Hotho, Knowledge Discovery on the Web). During the process of this algorithm, the data is first labelled and then trained using multi-level architectures (ensemble learning). The AdaBoost (Adaptive Boosting) is the result of this process (Freund, et al., 1995). Figure 15 gives an overview of the architecture of the boosting technique.

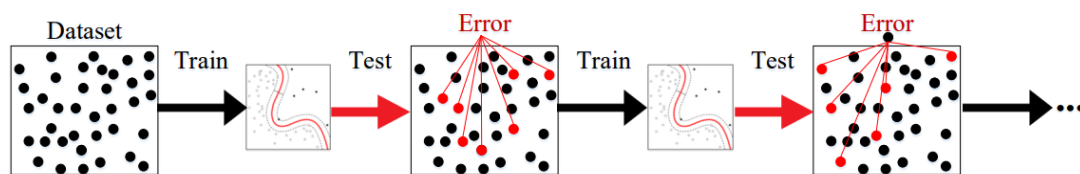


Figure 15: Boosting Technique Architecture (Kowsari, et al., 2019).

As mentioned previously, AdaBoost is the result of the Boosting process displayed in Figure 15. Using the training set S of size m an inducer τ and an integer N as input, the algorithm finds the weights of each x_j and the output is considered the optimal classifier C^* .

The final classifier formulation is as follows:

$$H_f(x) = \text{sign} \left(\sum_t \alpha_t h_t(x) \right)$$

Figure 16 gives an overview of the AdaBoost method's algorithm.

```

input : training set  $S$  of size  $m$ , inducer  $\tau$ , integer  $N$ 
for  $i = 1$  to  $N$  do
     $C_i = \tau(S')$ 

     $\epsilon_i = \frac{1}{m} \sum_{x_j \in S'; C_i(x_j) \neq y_j} \text{weight}(x)$ 

    if  $\epsilon_i > \frac{1}{2}$  then
        set  $S'$  to a bootstrap sample from  $S$  with weight 1 for
        all instance and go top
    endif

     $\beta_i = \frac{\epsilon_i}{1 - \epsilon_i}$ 
    for  $x_i \in S'$  do
        if  $C_i(x_j) = y_i$  then
             $\text{weight}(x_j) = \text{weight}(x_j) \cdot \beta_i$ 
        endif
    endfor
    Normalize weights of instances
endfor

 $C^*(x) = \arg \max_{y \in Y} \sum_{i, C_i(x)=y} \log \frac{1}{\beta_i}$ 

output: Classifier  $C^*$ 

```

Figure 16: The AdaBoost Method Algorithm (Kowsari, et al., 2019).

2.1.5.3. Bagging

First introduced in 1996 as a voting classifier method (Breiman, 1996), the bagging algorithm is “generated by different bootstrap samples” (Bauer & Kohavi, 1999). Bootstraps goal is to generate samples for the training set and if N bootstrap samples B_1, B_2, \dots, B_N are generated then there are N classifiers (C). Also, C_i is generated from each sample B_i . Figure 17 shows the bagging technique process while training N models.

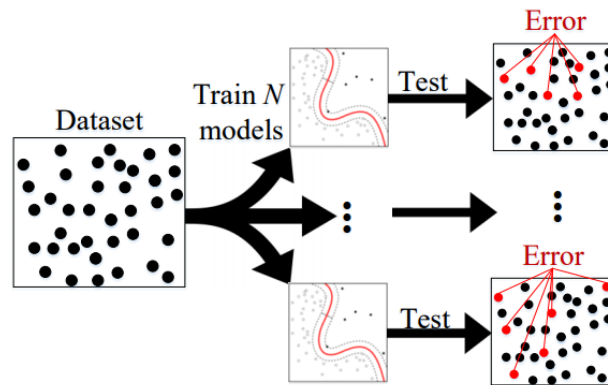


Figure 17: Simplified model of the bagging technique (Kowsari, et al., 2019).

Using the training set S , an inducer τ and an integer N as input, the algorithm finds the best classifier C . Gives an idea of how the bagging algorithm works.

```

input : training set  $S$ , inducer  $\tau$ , integer  $N$ 
for  $i = 1$  to  $N$  do
     $S' =$  bootstrap sample from  $S$ 
     $C_i = \tau(S')$ 
endfor

 $C^*(x) = \arg \max_{y \in Y} \sum_{i, C_i = y} 1$ 

output: Classifier  $C^*$ 

```

Figure 18: Bagging Algorithm (Kowsari, et al., 2019).

Limitations

Boosting and Bagging algorithms present many limitations and drawbacks. (Geurts, 2000) argues that one of the most noticeable limitations of those models is their “computational complexity and loss of interpretability”. In other words, both models do not explore features complexity in their process.

2.1.5.4. Logistic Regression (LR)

LR is one of the first classification methods to be proposed. It is a unique linear classifier that unlike other methods, focuses on the prediction of probabilities instead of classes. This section discusses three instances of LR: The Basic Framework model, the model combining Instance-Based Learning and LR and finally the Multinomial LR.

2.1.5.4.1. Basic Framework

The basic goal of LR is training “from the probability of a variable Y being 0 or 1 given a x ” (Kowsari, et al., 2019). In the case of a binary classification problem, (Juan & Vidal, 2002) argue that the Bernoulli mixture model functions are to be used as follows:

$$\begin{aligned}
 L(\theta | x) &= p(y | x; \theta) = \\
 &= \prod_{i=1}^n \beta(y_i | \text{sigm}(x_i \theta)) = \\
 &= \prod_{i=1}^n \text{sigm}(x_i)^{y_i} (1 - \text{sigm}(x_i))^{1-y_i} = \\
 &= \prod_{i=1}^n \left[\frac{1}{1 + e^{-x_i \theta}} \right]^{y_i} \left[1 - \frac{1}{1 + e^{-x_i \theta}} \right]^{(1-y_i)}
 \end{aligned} \tag{12}$$

In equation (12), $x_i\theta = \theta_0 + \sum_{j=1}^d(x_{ij}\theta_j)$ and to determine $\text{sigm}(\cdot)$, the following equation (13) is used:

$$\text{sigm}(\eta) = \frac{1}{1 + e^{-\eta}} = \frac{e^{\eta}}{1 + e^{\eta}} \quad (13)$$

2.1.5.4.2. Combination of Instance-Based Learning and LR

As previously mentioned, LR determines the probability of $y_i = \{0,1\}$ given x_i . The posterior probability can then be determined using equation (14):

$$\pi_0 = P(y_0 = +1|y_i) \quad (14)$$

With

$$\frac{\pi_0}{1 - \pi_0} = \frac{P(y_i|y_0 = +1)}{P(y_i|y_0 = -1)} \cdot \frac{p_0}{1 - p_0} \quad (15)$$

and p as the likelihood ratio that can also to be written as:

$$\frac{\pi_0}{1 - \pi_0} = p \cdot \frac{p_0}{1 - p_0} \quad (16)$$

$$\log\left(\frac{\pi_0}{1 - \pi_0}\right) = \log(p) + w_0 \quad (17)$$

Considering:

$$w_0 = \log(p_0) - \log(1 - p_0) \quad (18)$$

To combine with Instance-Based Learning (IBL), a few rules apply to the classifier. The classifier has to be a function of distance with δ_i . p large if $\delta_i \rightarrow 0$ and $y_i = +1$.

In case $y_i = -1$, the classifier is small and p close to 1 if $\delta_i \rightarrow \infty$. Finally, in case of neither $y_0 = -1$ nor $y_0 = +1$ the following function applies.

$$p = p(\delta) = \exp\left(y_i \cdot \frac{\alpha}{\delta}\right) \quad (19)$$

Finally,

$$\log\left(\frac{\pi_0}{1 - \pi_0}\right) = w_0 + \alpha \sum_{x_i \in N(x_0)} k(x_0, x_i) \cdot y_i \quad (20)$$

With $k(x_0, x_i)$ being the similarity measure.

Limitations

Although LR classifier has shown outstanding performance for the prediction of categorical outcomes, (Huang, 2015) argues that one of the limitations of LR is that it requires each data point to be independent. Furthermore, (Guerin, 2016) adds that the need for data points to be independent results in a prediction method based on independent parameters.

2.1.5.5. Naïve Bayes Classifier (NBC)

Naïve Bayes Classifiers can be classified as probabilistic classifiers used in Machine Learning. Mainly used for information retrieval, Naïve Bayes Classifiers basic goal is to determine the probability of a document D belonging to a class C . Although NBC main trait is its ability for probabilistic approximation, it also possesses the ability to learn like ML algorithms. This section is divided into three sections. The first section gives an advance description of NBC, the second reviews Multinomial NBC and the third NBC for Unbalanced Classes.

2.1.5.5.1. Advanced Description of NBC

In the Naïve Bayes algorithm, if the number of documents (n) fits into k categories with $k \in \{c_1, c_2, \dots, c_k\}$ then the predicted class is $k \in C$. In equation (21), d represents document and c indicates classes. The algorithm can be described as followed:

$$P(c|d) = \frac{P(d|c) P(c)}{P(d)} \quad (21)$$

with

$$C_{MAP} = \arg \max_{c \in C} P(d|c) P(c) \quad (22)$$

$$= \arg \max_{c \in C} P(x_1, x_2, \dots, x_n | c) p(c) \quad (23)$$

A word-level representation of NBC is shown in equation (24). This representation is used as a standard in many papers discussing and improving NBC.

$$P(c_j | d_i, \hat{\theta}) = \frac{P(c_j | \hat{\theta}) P(d_i | c_j, \hat{\theta}_j)}{P(d_i | \hat{\theta})} \quad (24)$$

2.1.5.5.2. Multinomial NBC

The multinomial NBC is a model mainly applied in case of large datasets; its methods have proven to provide excellent results in data pre-processing due to its outstanding computational abilities. Although NBC provides some advantages, (Mahender & Korde, 2012) identifies two significant problems. Its rough parameter estimation and its inability to handle rare categories with few training documents. Many researchers have proposed approaches to fixing some of those drawbacks.

To solve the rare category handling problem, (Kim, et al., 2006) proposes a weight enhancing method that functions by assigning different weight values to different attributes. (Kim, et al., 2006) also proposes the Poisson model who aims at improving NB classification performance. Another method proposed is the search of dependencies among attributes proposed by (Pazzani, 1996) to improve the NBC process.

In Multinomial NBC, if the number of documents (n) fits into k categories with $k \in \{c_1, c_2, \dots, c_k\}$ then the predicted class is $k \in C$. In equation (25), n_{wd} represents the number of times a word w appears in a document. $P(w|c)$ is for the probability of a word w occurring in a class c (Frank & Bouckaert, 2006) and it can be calculated using equation (26).

$$P(c|d) = \frac{P(c) \prod_{w \in d} P(w|c)^{n_{wd}}}{P(d)} \quad (25)$$

$$P(w|c) = \frac{1 + \sum_{d \in D_c} n_{wd}}{k + \sum_{w'} \sum_{d \in D_c} n_{w'd}} \quad (26)$$

2.1.5.5.3. Naïve Bayes Classifier for Unbalanced Classes

This section reviews a solution proposed to solve one of the main drawbacks of NBC: its poor performance with unbalanced classes (Liu, et al., 2009). (Frank & Bouckaert, 2006) proposes a two-step method to solve it. First, normalization is introduced in each class by using equation (27) then the centroid classifier is used for unbalanced classes. The centroid C_c for a given class c is provided in equation (28).

$$\propto \times \frac{n_{wd}}{\sum_{w'} \sum_{d \in D_c} n_{w'd}} \quad (27)$$

$$C_c = \left\{ \frac{\sum_{d \in D_c} n_{w_1 d}}{\sqrt{\sum_w (\sum_{d \in D_c} n_{wd})^2}}, \dots, \frac{\sum_{d \in D_c} n_{w_i d}}{\sqrt{\sum_w (\sum_{d \in D_c} n_{wd})^2}}, \dots, \frac{\sum_{d \in D_c} n_{w_k d}}{\sqrt{\sum_w (\sum_{d \in D_c} n_{wd})^2}} \right\} \quad (28)$$

The scoring function can be determined using equation (29):

$$x_d \cdot c_1 - x_d \cdot c_2 \quad (29)$$

So, Log of NBC can be determined by using equation (30).

$$\left[\log P(c_1) + \sum_{i=1}^k n_{w_i d} \log(P(w_i | c_1)) \right] - \left[\log P(c_2) + \sum_{i=1}^k n_{w_i d} \log(P(w_i | c_2)) \right] \quad (30)$$

By using equation (26) and (27), $P(w|c)$'s formula can be updated to equation (31) if $\alpha = 1$

$$P(w|c) = \frac{1 + \frac{n_{wd}}{\sum_{w'} \sum_{d \in D_c} n_{w'd}}}{k + 1} \quad (31)$$

And

$$\frac{\sum_{d \in D_c} n_{wd}}{\sum_{w'} \sum_{d \in D_c} n_{w'd}} \ll 1 \quad (31)$$

Considering $\log(x + 1) \approx x$ and $x \ll 1$ (Frank & Bouckaert, 2006). The result of this process is similar to the one in the centroid classifier experiment in (Han & Karypis, 2000).

Limitations

NBC has various limitations. (Liu, et al., 2009) mentions that one of the main limitations of NBC is its poor results when applied to datasets with unbalanced classes.

(Soheily-Khah, et al., 2018) and (Wang, et al., 2016) on the other hand identify one of NBC's limitation to be the fact that it has strong assumptions about the shape of data distribution. Finally, (Ranjan, et al., 2017) states that "NBC is also limited by data scarcity for which any possible value in feature space, a likelihood value must be estimated by a frequentist".

2.1.5.6. K-Nearest Neighbor (KNN)

The K-Nearest Neighbor algorithm is a "non-parametric technique used for classification". It works on the principle of closest training samples. In other words, data points close to one other are automatically grouped in the same class. This process is known as instance-based learning (Nidhi & Gupta, 2011). In this section, we discuss two concepts of the K-Nearest Neighbor: the basic concept and the Weight Adjustment K-Nearest Neighbor Classification.

2.1.5.6.1. Basic Concept of KNN

The KNN algorithm performs two basic tasks. First, it identifies k nearest neighbors of x among all documents in the training set with x being a test document and gives scores to the category's candidates based on the class of the k nearest neighbors identified. Once the scores are allocated, the candidates are assigned to the class with the highest score identified from the document x . KNN decision rule can be formulated as such:

$$\begin{aligned} f(x) &= \arg \max_j S(x, C_j) \\ &= \sum_{d_i \in KNN} sim(x, d_i) y(d_i, C_j) \end{aligned} \quad (32)$$

And S being the score value relating to $S(x, C_j)$, i being the candidate, j the class and $f(x)$ the label of the set of test documents.

Figure 19 gives a graphical representation of the architecture of the KNN model for a 2D data set with three classes (Kowsari, et al., 2019). It is observed that the document x introduced are assigned to the Neighbor class with the highest score (represented as blue triangles and red circles).

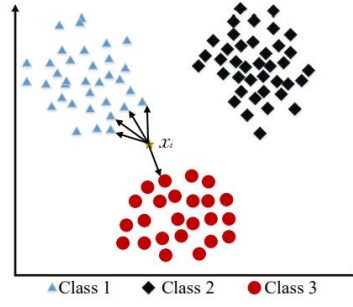


Figure 19: Architecture of KNN for a 2D data set and Three classes (Kowsari, et al., 2019).

2.1.5.6.2. Weight Adjustment K-Nearest Neighbor Classification (WAKNN)

The WAKNN is an advanced version of KNN that aims at learning and introducing weight vectors in the classification process (Han, et al., 2001). To calculate the value of the weighted cosine, equation (33) is used (Salton, 1989):

$$\cos(x, y, w) = \frac{\sum_{t \in T} (x_t \times w_t) \times (y_t \times w_t)}{\sqrt{\sum_{t \in T} (x_t \times w_t)^2} \times \sqrt{\sum_{t \in T} (y_t \times w_t)^2}} \quad (33)$$

With T representing the set of words, x_t and y_t as TF. Also, $N_d = \{n_1, n_2, \dots, n_k\}$ stands for the set of k-nearest neighbors d . With N_d identified, the sum of the neighbors d belonging in a class c can be determined as follows:

$$S_c = \sum_{n_i \in N; C(N_i)=c} \cos(d, n_i, w) \quad (34)$$

The total similarity is further calculated using equation (35).

$$T = \sum_{c \in C} S_c \quad (35)$$

Finally, the contribution of d referred to as $cont(d)$ is defined based on the similarity sum S_c and T as follows:

$$cont(d) = \begin{cases} 1 & \text{if } \forall c \in C, c \neq class(d), \\ & S_{class(d)} > S_s \text{ and } \frac{S_{class(d)}}{T} \leq p \\ 0 & \text{otherwise} \end{cases} \quad (36)$$

Limitations

Although KNN is an algorithm widely adopted due to its effectiveness, its robustness when dealing with noisy data and its ability to adapt to any kind of feature space, it also has various limitations. (Kowsari, et al., 2019) argue that KNN is limited by data storage thus cannot be used for large search problems for the identification of nearest neighbors.

(Sahgal & Parida, 2014) and (Sanjay, et al., 2018) claim that since KNN is the type of algorithm that finds meaningful distance, it is an algorithm too data-dependent.

In an attempt to contrast KNN advantages and limitation, (Aramaki & Miyo, 2006) identifies one of KNN's limitation to be its long computational time and the difficulty posed by finding an optimal value for k . Many methods exist to improve those limitations such as a Tree-Based KNN proposed by (Maillo, 2017), distributed techniques such as MapReduce to reduce memory constraints, the use of semi-supervised learning mechanism proposed by (Thangaraj & Sivakami, n.d.).

2.1.5.7. Support Vector Machine (SVM)

First introduced in 1963 by (Vapnik & Chervonenkis, 1964), SVM was introduced for binary classification tasks. Many researchers later worked on SVM to improve its processes and performance. (Boser, et al., 1992) later adapts the original version and introduces a nonlinear formulation. SVM's key principle is to partition the data space by using linear or non-linear delineations to establish the optimal boundaries between the classes. The boundaries are then used to perform efficient text classification. SVM classifiers are known for their ability to improve text classification and combined with other methods such as Hidden Markov Models (HMM) used for feature extraction, SVM's capabilities are improved greatly allowing it to successfully classify unknown text (Wang, et al., 2007). In the same context, combined with Naïve Bayes algorithms with Naïve Bayes in charge of reducing the number of features (Lin, 2002), it was noticed that SVM was capable of solving multi-label class classification (Qin & Wang, 2009). In this section two techniques/variation of SVM are reviewed: String Kernel and Multi-Instance Learning (MIL) and some SVM limitations are also discussed.

2.1.5.7.1. String Kernel

The basic principle of String Kernel is to “map the string in the feature space” using the feature map $\Phi(\cdot)$ (Kowsari, et al., 2019). In this method, a variation called Spectrum kernel is used. Spectrum Kernel identifies the number of times a word occurred in a string x_i as a feature map with feature maps being defined as $x \rightarrow R^{l^k}$.

$$\Phi_k(x) = \Phi_j(x)_{j \in \Sigma_k} \quad (37)$$

where

$$\Phi_j(x) = \text{number of } j \text{ feature appears in } x \quad (38)$$

The feature map $\Phi_i(x)$ is then generated by the sequence x_i and the kernel is defined as follows:

$$F = \sum k \quad (39)$$

$$K_i(x, x') = \langle \Phi_i(x), \Phi_i(x') \rangle \quad (40)$$

In equation (39), the features are created using a size Σ dictionary and F is the number of features. The kernel is then calculated using equation (40) and normalized using equation (41).

$$K^{Norm}(x, y) \leftarrow \frac{K(x, y)}{\sqrt{K(x, x)} \sqrt{K(y, y)}} \quad (41)$$

$$\langle f^x, f^y \rangle = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} h(u_i^{s_1}, u_j^{s_2}) \quad (42)$$

Where two sequences: $u_i^{s_1}$ and $u_j^{s_2}$ are lengths of s_1 and s_2 respectively. This method is further developed and applied in sectors such as protein classification and DNA (Leslie, et al., 2002) (Leslie, et al., 2004).

2.1.5.7.2. Multi-Instance Learning (MIL)

Multi-Instance learning (MIL) can be classified as an SVM based supervised learning method. Unlike other methods, MIL receives sets of labelled bags containing various instances as input instead of individually marked instances. The bags can be either positive or negative following simple rules. A bag is labelled negative if all instances of it are negative and labelled positive if a least one instance in it is positive.

From the inputs, the learner has two basic choices. First, initiate a concept capable of labelling individual instances successfully or efficiently labelling bags without inducing the concept (Maron & Lozano-Pérez, 1997). To better understand this method, let us review it using a statistical pattern recognition process.

Considering a training set of labelled patterns with each pair $(x_i, y_i) \in R^d \times Y$ generated independently from an unknown distribution, the aim is to determine a classifier that works from patterns to labels: $f : R^d \rightarrow Y$. For the MIL algorithm, the input is introduced as a set of input patterns under the following format: x_1, \dots, x_n and are grouped into bags B_1, \dots, B_m where $B_I = \{x_i : i \in I\}$ for an index set as follows: $I \subseteq \{1, \dots, n\}$.

Also, each bag is associated with a label Y_I where $Y_I = -1$ if $y_i = -1$ for all $i \in I$ and $Y_I = 1$ if at least one instance $x_i \in B_I$ is labelled positive.

The relationship between instance labels y_i and bag labels Y_I is expressed in equation (43) and further described in equation (44) as a set of linear constraints.

$$Y_I = \max_{i \in I} y_i \quad (43)$$

$$\sum_{i \in I} \frac{y_i + 1}{2} \geq 1,$$

$$\forall I \text{ s. t. } Y_I = 1, \quad (44)$$

$$y_i = -1, \forall I \text{ s. t. } Y_I = -1.$$

Limitations

In term of machine learning capabilities, SVM has an efficient algorithm. However, its performance for text classification is not of the same stature. (Karamizadeh, et al., 2014) argue that one of SVM's limitation is its "lack of transparency in results caused by a high number of dimensions".

This results in the algorithm being unable to present scores in a parametric function form or other function forms (Karamizadeh, et al., 2014). Another limitation of SVM is a variable financial ratios rate (G., 2014).

2.1.5.8. Decision Tree

Decision Tree is a classification algorithm mainly used for text and data mining. Introduced by D. Morgan and developed by J.R Quinlan, it functions by performing a "hierarchical decomposition of the (training) data space".

To achieve that, a ***predicate*** or condition is used on the attribute value and is used to accurately divide the data space hierarchically. In case of data classification, the predicate is usually a condition related to the presence or absence of one or more words in a document and the data space division is done recursively until a certain class purity is met or leaf nodes contains a certain number of records (Aggarwal C.C., 2012). To simplify the process, Decision Tree can be viewed as a method that uses attributes to create a tree that can categorize data points. From this explanation, one of the big difficulties of Decision Tree can be deduced: which features, or attributes are to be taken to the parent's level and which are to be assigned for the child level.

To resolve this, (De Mántaras, 1991) proposes a method using statistical modelling to select features in a tree. To determine the fittest as the parent's node the following process is used. First, for a training set containing p positive and n negative:

$$H\left(\frac{P-N}{P+N}\right) = -\frac{p}{n+p} \log_2 \frac{p}{n+p} - \frac{n}{n+p} \log_2 \frac{p}{n+p} \quad (45)$$

Attribute A is chosen with k as distinct value and the training set E is divided into subsets of $\{E_1, E_2, \dots, E_k\}$. "The expected entropy (EH) remain after trying attribute A " (with branches $i = 1, 2, \dots, k$):

$$EH(A) = \sum_{i=1}^K \frac{p_i + n_i}{p + n} H\left(\frac{p_i}{n_i + p_i}, \frac{n_i}{n_i + p_i}\right) \quad (46)$$

Information gain (I) or "reduction in entropy for this attribute is" estimated following equation (46) and attribute with the largest information gain is chosen as the parent's node.

$$A(I) = H\left(\frac{p}{n+p}, \frac{n}{n+p}\right) - EH(A) \quad (47)$$

Limitations

As previously discussed, Decision Tree is an efficient algorithm when it comes to learning and prediction. However, it has critical limitations such as its hyper sensibility to perturbations in the data (Giovanelli, et al., 2017) and its problems with "out-of-sample" prediction (Jasim, 2016). (Quinlan, 1987) argues that one of the limitations observe in the Decision Tree is its tendency to be easily overfit resulting in the classifier

becoming inefficient due to the swapping of training tuples. To solve this, (Mehta, et al., 1996) proposes a method capable of handling numeric and categorical data. Many other researchers propose methods and techniques to improve on the Decision-Tree algorithm.

(Vateekul & Kubat, 2009) proposes a Fast Decision-Tree (FDT) induction method that can handle multi-label documents and reduce the cost of induction while (Johnson, et al., 2002) suggests the use of a decision-tree-based symbolic rule for text categorization and (Lewis & Ringuette, 1994) which proposes a method that provides great results when combined with decision support tools.

2.1.5.9. Random Tree

Random forest is “an ensemble learning method for text classification” (Kowsari, et al., 2019). This method aims at generating random decision trees to facilitate the classification process. Introduced in 1995 by T. Kam Ho (Quinlan, 1987), Random Forest uses trees t as parallel. This method was further developed in 1999 by L. Breiman who proposed convergence for Random Forest as margin measures ($mg(X, Y)$) as described in equation (48) with $I(.)$ being the indicator function. Figure 20 gives an overview of the random forest process.

$$mg(X, Y) = av_k I(h_k(X) = Y) - \max_{j \neq Y} av_k I(h_k(X) = j) \quad (48)$$

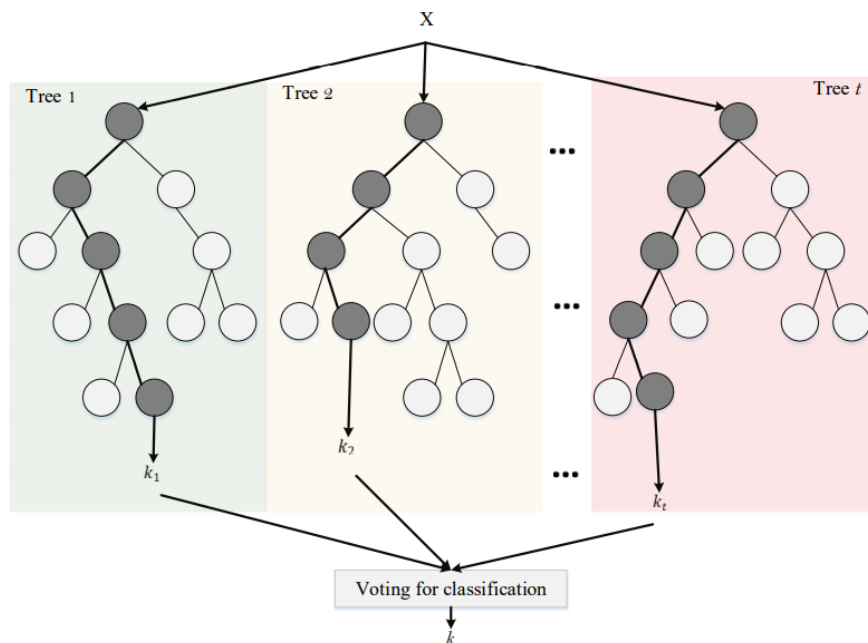


Figure 20: Random Forest (Kowsari, et al., 2019).

At the end of this process, random trees are trained and turned into a forest. The prediction is then assigned based on voting (Breiman, 2001) as displayed in equation (49).

$$\delta_V = \arg \max_i \sum_{j:j \neq i} I_{\{r_{ij} > r_{ji}\}} \quad (49)$$

Such that:

$$r_{ij} + r_{ji} = 1 \quad (50)$$

Limitations

(Bansal, et al., 2018) argues that although Random Forest is fast when it comes to the training of text-based datasets, it is very slow to generate predictions once trained. (Kowsari, et al., 2019) claims that by reducing the number of trees in the forest, the problem of time complexity during the prediction step can be solved.

2.1.5.10. Deep Learning

Deep learning methods/algorithms have been used in various domains including text classification and have proven to be extremely efficient. In TC, deep learning methods take multiple architectures. In this section, three deep learning architectures are reviewed: Deep Neural Networks (DNN), Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN).

2.1.5.10.1. Deep Neural Networks (DNN)

The DNN architecture functions such as “each layer only receive input from the previous layer and output to the next layer” (Kowsari, et al., 2017). Layers in DNN are fully connected and the input layer is composed of text features while the output layer consists of nodes for each classification label or in case of binary classification, consist of a single node. The input here is a string obtained as a process of vectorization of primary data.

The input layer can be built using advanced extraction methods such as TF-IDF, word embedding and other well-known methods. The output layer on the other hand equates to the total number of classes used for the classification (binary class or multi-class). For the multi-class DNNs, learning models are generated by randomly assigning the number of layers and the number of nodes each layer contains.

DNN is implemented by using “a discriminatively trained model that uses a standard back-propagation algorithm using sigmoid” (Kowsari, et al., 2019). Equation (51) gives an overview of the sigmoid equation while equation (52) is used as an activation function.

$$f(x) = \frac{1}{1 + e^{-x}} \in (0,1) \quad (51)$$

$$f(x) = \max(0, x) \quad (52)$$

The output layer in the case of multi-class classification is also determined following equation (53).

Figure 21 gives an overview of a standard fully connected DNN given a set (x, y) with $x \in X$ and $y \in Y$. To further simplify the process, it could be said that the aim is to learn the relation between the input and identify spaces using hidden layers.

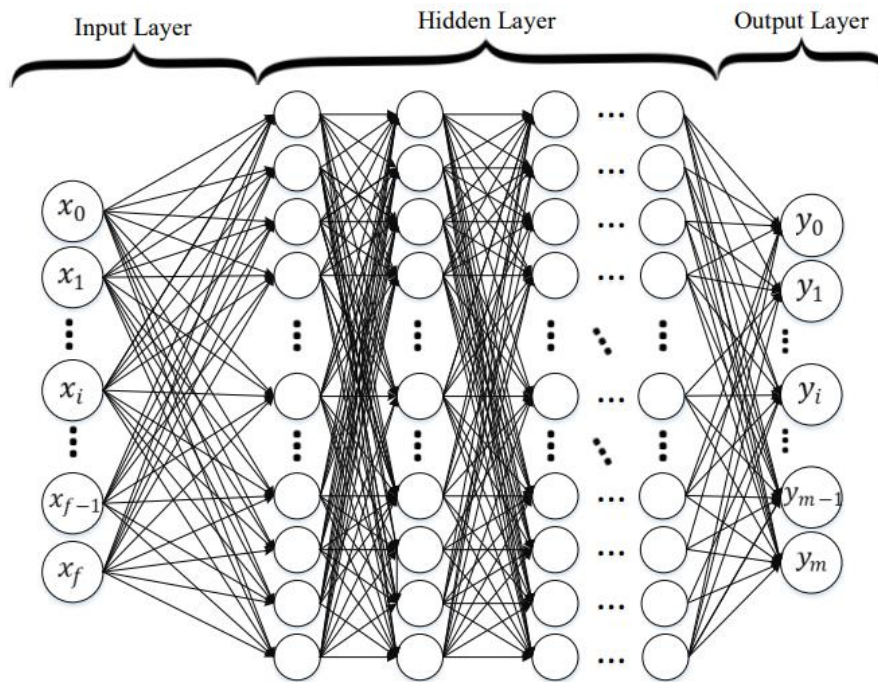


Figure 21: Standard, fully connected Deep Neural Network (DNN) (Kowsari, et al., 2019).

2.1.5.10.2. Recurrent Neural Network (RNN)

In the Recurrent Neural Network (RNN) architecture unlike in DNN, the output from a layer can be reinserted as input to that layer. In other words, DNN “assigns more weight to previous data points of a sequence”.

By using this process, RNN analyses information from previous nodes by using advance techniques such as LSTM and GRU further discussed in this section, making an excellent method to use for text processing. Equation (53) gives an overview of the RN method using a formula:

$$x_t = F(x_{t-1}, u_t, \theta) \quad (53)$$

Where x_t is the state at time t and u_t the input at step t . Equation (53) can be further detailed using weights with W_{rec} the recurrent matrix weight, W_{in} the input weights, b the bias and σ an element-wide function.

$$x_t = W_{rec}\sigma(x_{t-1}) + W_{in}u_t + b \quad (54)$$

Figure 22 gives an overview of RNN using the LSTM and GRU techniques for Text Classification.

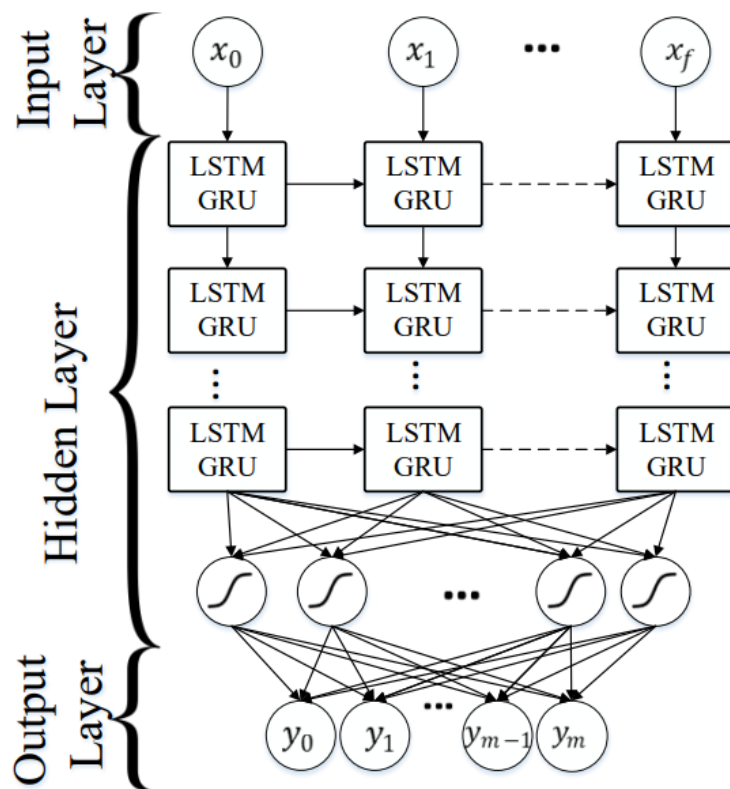


Figure 22: Standard LSTM/GRU Recurrent Neural Networks (RNN) (Kowsari, et al., 2019).

▪ Long Short-Term Memory (LSTM)

LSTM is a type of RNN designed to fix the problem of vanishing gradients faced by the standard RNN (Kowsari, et al., 2017). Although LSTM has a lot of similarities with RNN,

it uses various gates to control the amount of information entering in each node state. Figure 23 gives an overview of an LSTM cell.

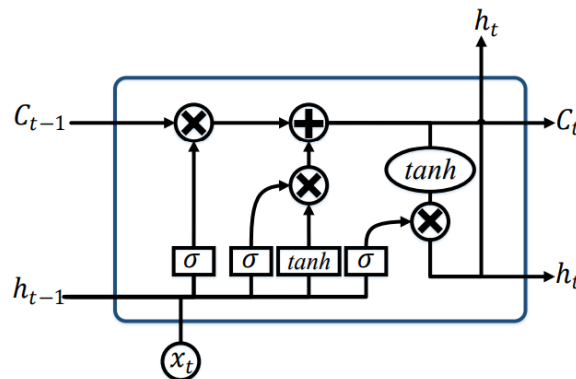


Figure 23: LSTM cell (Kowsari, et al., 2017).

As previously mentioned, LSTM regulates the amount of information in each node state using 5 gates: **(1) Input Gate**, **(2) Candid Memory Cell Value**, **(3) Forget Activation**, **(4) New Memory Cell Value** and **(5) Output Gate Values** respectively indicated in equation (55) to (60).

$$(1) \quad i_t = \sigma(W_i[x_t, h_{t-1}] + b_i), \quad (55)$$

$$(2) \quad \tilde{C}_t = \tanh(W_c[x_t, h_{t-1}] + b_c), \quad (56)$$

$$(3) \quad f_t = \sigma(W_f[x_t, h_{t-1}] + b_f), \quad (57)$$

$$(4) \quad C_t = i_t * \tilde{C}_t + f_t C_{t-1}, \quad (58)$$

$$(5) \quad o_t = \sigma(W_o[x_t, h_{t-1}] + b_o), \quad (59)$$

$$(5) \quad h_t = o_t \tanh(C_t), \quad (60)$$

In the previous equations, W represents the weight matrix, x_t the input of memory cell at time t and i, c, f, o indicate respectively the input, cell memory, forget and output gates.

▪ Gated Recurrent Unit (GRU)

GRU is a gating mechanism and a simplified version of LSTM. It follows the same architecture however it is different on two levels. First, unlike LSTM, GRU has two gates and those gates do not possess an internal memory. Next, non-linearity is not included in GRU.

The GRU model was introduced as a mean to solve the problem of RNN being biased in the presence of newer influential words. Figure 24 gives an overview of a GRU cell.

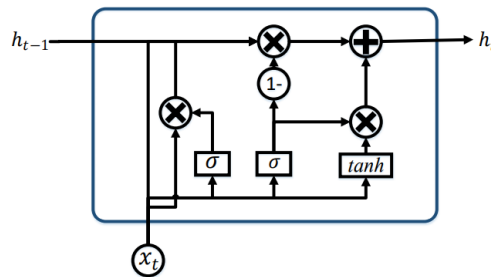


Figure 24: GRU Cell (Kowsari, et al., 2017).

Limitation

Despite the aforementioned advantages, (Bengio, et al., 1994) identifies one of RNN vulnerabilities to be “problems of vanishing gradients and exploding gradient when the error of the gradient descent algorithm is backpropagated through the network”. RNN can also be biased when recent words are added.

2.1.5.10.3. Convolutional Neural Network (CNN)

CNN is a deep learning method mostly used in text classification for hierarchical document classification (Lai, et al., 2015) (Jaderberg, et al., 2015). First introduced for image processing, it has since been improved and used as an efficient text classification method.

In a typical CNN for image processing, an image tensor is combined with a set of kernels of size $d \times d$. The layers resulting from this combination are called feature maps and can be staked to improve filtering on the input. As for the output, pooling methods are generally used to reduce the output from one layer to the next while preserving critical features needed for the rest of the process. A common method used is the max pooling method. Its process is simple, the output with the highest pool is selected and can proceed to the next layer. However, before the pooled output can proceed to the next layer, the feature maps are flattened into a single column. Furthermore, weights and feature detector filters are adjusted throughout the back-propagation process to prevent unforeseen errors.

Limitation

As previously mentioned, CNN was originally designed for image processing meaning, it is not a perfect method for text classification. (Kowsari, et al., 2017) identifies one of

CNN limitations to be the number of channels or size of feature space. Ideal for images (3 channels) it can become quite large in the context of “text processing”.

2.1.6. Training

At this stage, classifiers need to be trained to efficiently perform their task. The most crucial element here is the dataset. (MonkeyLearn, n.d.) mentions that “a text classifier is worthless without accurate data training to power it”. The training stage can be summarized to: “telling the algorithm what type of tags or set of tags it should output when a document is inputted”. The algorithm then learns to recognise the pattern in the text and with appropriate dataset and training, is capable of predicting intent, sentiment and much more.

In this section, some hate speech datasets are reviewed based on availability, class, size, and format. Next, an analysis of previous work related to the detection of abusive language is also done taking into consideration the approach used.

During a typical training stage, the data is divided into two: the training set and testing set. The training set is used to train the classifier and the testing set is used to test the ability of the algorithm to predict efficiently when data it never encountered is feed to it.

2.1.6.1. Datasets

Datasets are critical for the training stage. While picking a dataset, the aim is to get the dataset that best represents the type of outcome the algorithm is supposed to predict. Table 5 provides an analysis of some hate speech datasets used and proposed by some researchers.

Table 5: Analysis of some of the existing hate speech datasets (Madukwe, et al., 2020).

Datasets	Availability	Class/Labels	Size	Format
BURNAP Dataset (Burnap & Williams, 2016)	No	Sexual Orientation Race Disability Religion	-	-
WASEEM Dataset	Yes	Racism	11.69%	TweetID

(Waseem & Hovy, 2016)		Sexism	20.00%	
		Neither	68.33%	
	16,914 tweets			
DAVIDSON Dataset (Davidson, et al., 2017)	Yes	Hate Speech	5.77%	Raw Text
		Offensive	77.43%	
		Neither	16.80%	
		24k tweets		
FOUNTA Dataset (Founta, et al., 2018)	Yes	Abusive	11%	TweetID
		Hateful	7.5%	
		Spam	22.5%	
		Normal/Noe	59%	
		80k tweets		
WARNER Dataset (Warner & Hirschberg, 2012)	No	Anti-Semitic		
		Anti-Black		
		Anti-Asian		
		Anti-Woman	-	-
		Anti-Muslim		
		Anti-Immigrant		
		Other hate		
DJURIC Dataset (Djuric, et al., 2015)	No	Hate Speech	5.91%	-
		Clean	94.08%	-
		951,736 comments		
NOBATA Dataset (Nobata, et al., 2016)	No	Abusive	7% of F + 16.4% of N	-
		Clean	3.4% of F + 10.7% of N	-
QIAN Dataset (Qian, et al., 2019)	Yes	Hate Speech	23.5%	Raw Text
		Non-Hate Speech	76.5%	
		22,324 Reddit comments		
QIAN Dataset (Qian, et al., 2019)	Yes	Hate Speech	43.2%	Raw Text
		Non-Hate Speech	51.8%	
		33,776 Gab comments		

ROSS Dataset (Ross, et al., 2016)	Yes	6 Point Likert Scale	- 541 tweets	-
BENIKOVA Dataset (Benikova, et al., 2018)	Yes	Hate Speech Non-Hate Speech	33% 67% 36 tweets	-
VIGNA Dataset (Del Vigna, et al., 2017)	No	No Hate Weak Hate Strong Hate	- - 17,567 Facebook comments	-
TULKENS Dataset (Tulkens, et al., 2016)	No	Racist Non-Racist	- 17,567 Facebook comments	-

Although datasets are critical, the approach used to get the best off the datasets is also important. Table 6 gives an overview of some of the previous work done for the detection of abusive language and takes a closer look at the approach adopted.

Table 6: Hate speech detection Approach (Pamungkas, et al., 2020).

Authors	Approach
Pamungkas (Pamungkas, et al., 2018)	"SVM with a combination of handcrafted stylistic, structural, and lexical features".
Goenaga (Goenaga, et al., 2018)	"Bi-LSTM with pre-trained word embeddings".
Liu (Liu, et al., 2018)	"Average probability of two traditional classifiers trained on doc2vec".
Canós (Canós, 2018)	"SVM with tf-idf unigrams".
Nina-Alcocer (Nina-Alcocer, 2018)	"SVM, Multi-layer Perceptron (MLP) and Multinomial Naive Bayes, with structural, lexical, and syntactical features".

Shushkevich (Shushkevich & Cardiff, 2018)	Logistic regression, naive Bayes, SVM, and ensemble classifier, with tf-idf.
Frenda (Frenda, et al., 2018)	"Ensemble of SVM classifiers with character n-grams, sentiment, and lexicons".
Pamungkas (amungkas, et al., 2018)	"Linear and RBF kernel SVM with structural and lexical features, including a multilingual hate lexicon".
Bakarov (Bakarov, 2018)	"Single Value Decomposition and boosting classifier with tf-idf".
Basile (Basile & Rubagotti, 2018)	"SVM with n-grams and cross-lingual classification with bleaching".
Saha (Saha, et al., 2018)	"Logistic regression trained on concatenated sentence embeddings, tf-idf, and average word embeddings".
Ahluwalia (Ahluwalia, et al., 2018)	"Voting ensemble with handcrafted features".
Shushkevich (Shushkevich & Cardiff, 2018)	"Ensemble of logistic regression, SVM, and naive Bayes, with tf-idf".
Buscaldi (Buscaldi & Davide, 2018)	"Bi-LSTM with character embedding and random forest with weighted n-grams".
Frenda (Frenda, et al., 2018)	"SVM and random forest with stylistic and lexical features and lexicons".

III. Summary of Techniques and Classifiers

In this section, a simple comparison of features extraction methods is first done, followed by a discussion on the advantages and limitations of the text classification methods described and finally a comparison of various text classification techniques is also done.

1. Feature Extraction Comparison

In this section, some feature extraction methods are discussed reviewing their advantages and limitation. The analysis is done in Table 7:

Table 7: Feature Extraction Comparison (Kowsari, et al., 2019).

Model	Advantages	Limitations
Weighted Words	<ol style="list-style-type: none">1. Easy to compute.2. Easily used to compute the similarity between 2 documents.3. Basic metric to extract the most descriptive terms in a document.4. Works with an unknown word.	<ol style="list-style-type: none">1. Does not capture the position in the text (syntactic).2. Does not capture meaning in the text (semantics).3. Common words effect on the results (e.g., “am”, “is”, etc.).
Word2Vec	<ol style="list-style-type: none">1. Captures position of the words in the text (syntactic).2. Captures meaning in the words (semantics).	<ol style="list-style-type: none">1. Cannot capture the meaning of the word from the text (fails to capture polysemy).2. It cannot capture out-of-vocabulary words from corpus.
FastText	<ol style="list-style-type: none">1. Works for rare words (rare in their character n-grams which are still shared with other words).2. Solves out of vocabulary words with n-gram in character level.	<ol style="list-style-type: none">1. Cannot capture the meaning of the word from the text (fails to capture polysemy).2. Memory consumption for storage.3. Its computation is more expensive than GloVe and Word2Vec.
TF-IDF	<ol style="list-style-type: none">1. Easy to compute.2. Easily used to compute the similarity between 2 documents.3. Basic metric to extract the most descriptive terms in a document.4. Common words do not affect the results (e.g., “am”, “is”, etc.).	<ol style="list-style-type: none">1. Does not capture the position in the text (syntactic).2. Does not capture meaning in the text (semantics).
Contextualized Word Representations	<ol style="list-style-type: none">1. Captures the meaning of the word from the text (incorporates context, handling polysemy).	<ol style="list-style-type: none">1. Memory consumption for storage.2. Cannot capture out-of-vocabulary words from corpus.3. Needs another word embedding for all LSTM and feed-forward layers.4. Works only sentence and document level (it cannot work for individual word level).

2. Text Classification Comparison

In this section, some feature extraction methods are discussed reviewing their advantages and limitation. The analysis is done in Table 8.

Table 8: Text Classification Comparison (Kowsari, et al., 2019).

Model	Advantages	Disadvantages
Rocchio Algorithm	<ol style="list-style-type: none"> 1. Easy to implement. 2. Does not require many computational resources. 3. Relevance feedback mechanism (benefits to ranking documents as not relevant). 	<ol style="list-style-type: none"> 1. Can only retrieve a few relevant documents. 2. Techniques not very robust. 3. Linear combination is not good for multi-class datasets.
Boosting and Bagging	<ol style="list-style-type: none"> 1. Improves stability and accuracy taking advantage of ensemble learning. 2. Reduces variance hence eliminates the risk of overfitting. 	<ol style="list-style-type: none"> 1. Computational Complexity. 2. Requires careful tuning of different hyper-parameters. 3. Loss of interpretability.
Logistic Regression	<ol style="list-style-type: none"> 1. Easy to implement. 2. Does not require many computational resources. 3. Does not require pre-processing. 4. Does not require any tuning. 	<ol style="list-style-type: none"> 1. Cannot solve linear problems. 2. Requires for each data point to be independent. 3. Prediction is based on independent variables.
Naïve Bayes Classifiers	<ol style="list-style-type: none"> 1. Works well with text datasets. 2. Easy to implement. 3. Faster than other algorithms. 	<ol style="list-style-type: none"> 1. Has a strong assumption about the shape of data distribution. 2. Limited by data scarcity.
K-Nearest Neighbor	<ol style="list-style-type: none"> 1. Works well with text datasets. 2. Non-parametric. 3. Naturally handles multi-class datasets. 	<ol style="list-style-type: none"> 1. Requires a lot of computational resources. 2. Difficult to get K's optimal value. 3. Difficulty in large scale searches for nearest neighbours.
Support Vector Machine (SVM)	<ol style="list-style-type: none"> 1. Capable of modelling non-linear decision boundaries. 2. Performs similarly to logistic regression when linear separation. 3. Robust against overfitting problems. 	<ol style="list-style-type: none"> 1. Lack of transparency in results caused by a high number of dimensions. 2. Memory complexity. 3. Difficulty in choosing an efficient kernel function.
Decision Tree	<ol style="list-style-type: none"> 1. Can easily handle qualitative features. 2. Performs similarly to logistic regression when linear separation. 3. Very fast algorithm for both learning and prediction. 	<ol style="list-style-type: none"> 1. Issues with diagonal decision boundaries. 2. Can be easily overfitted. 3. Problems with out-of-sample prediction.
Random Forest	<ol style="list-style-type: none"> 1. Fast training process. 2. Reduced variance. 3. Does not require pre-processing and preparation of the input data. 	<ol style="list-style-type: none"> 1. Slow to make predictions. 2. More trees increase time complexity in the prediction step. 3. Not easy to visually interpret. 4. Can easily overfit.
Deep Learning	<ol style="list-style-type: none"> 1. Flexible with features design. 2. Architecture that can be adapted to new problems. 3. Can deal with complex input-output mappings. 4. Can easily handle online learning. 5. Parallel processing capability. 	<ol style="list-style-type: none"> 1. Requires a large amount of data to outperform other approaches. 2. Is extremely computationally expensive to train. 3. Finding an efficient architecture and structure is still the main challenge of this technique.

IV. Methodology

1. Research Process

The research process refers to the procedure used by researchers to build and write a viable research paper. It usually involves identifying, locating, assessing, and analyzing various information related to the research. Figure 25 gives an overview of the research flowchart.

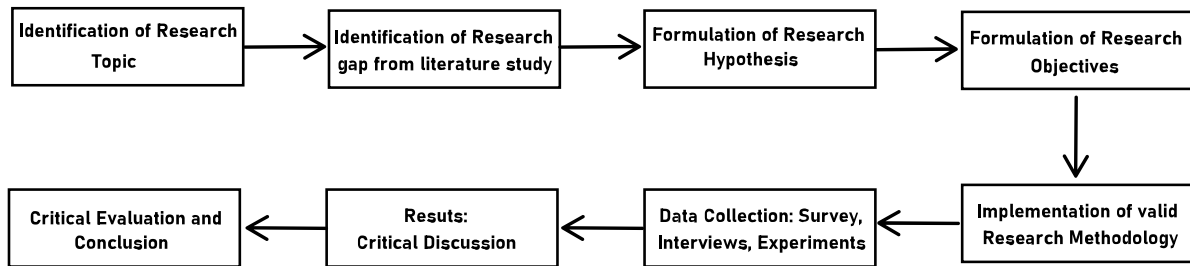


Figure 25: Research process Flowchart.

2. Research Methodology

Research methodology refers to the systematic analysis of the methods applied to a field of study. In other words, it is the stage where the researcher takes into consideration the different methods and models that will be used for the research. Research methodology usually encompasses paradigm, theoretical model, phases and quantitative or qualitative techniques (Irny & Rose, 2005).

For this research, several models are used: the quantitative and qualitative techniques are used to analyze the available datasets necessary for the training of the classifiers. As previously mentioned, the dataset selected impacts greatly the performance of the classifier. The dataset needs to be of great quality for the feature extraction process and moderate in quantity for the algorithm to train efficiently. To that end, two models are used for this research: the systematic literature review method used to produce a detailed and informative literature review about classifiers and existing datasets and the research onion method used to collect experimental data.

An efficient way to build a strong research methodology is using the “research onion” proposed by Saunders. It provides detailed guidance on the layers and stages to achieve a capable methodology.

Figure 26 summarizes the research onion model.

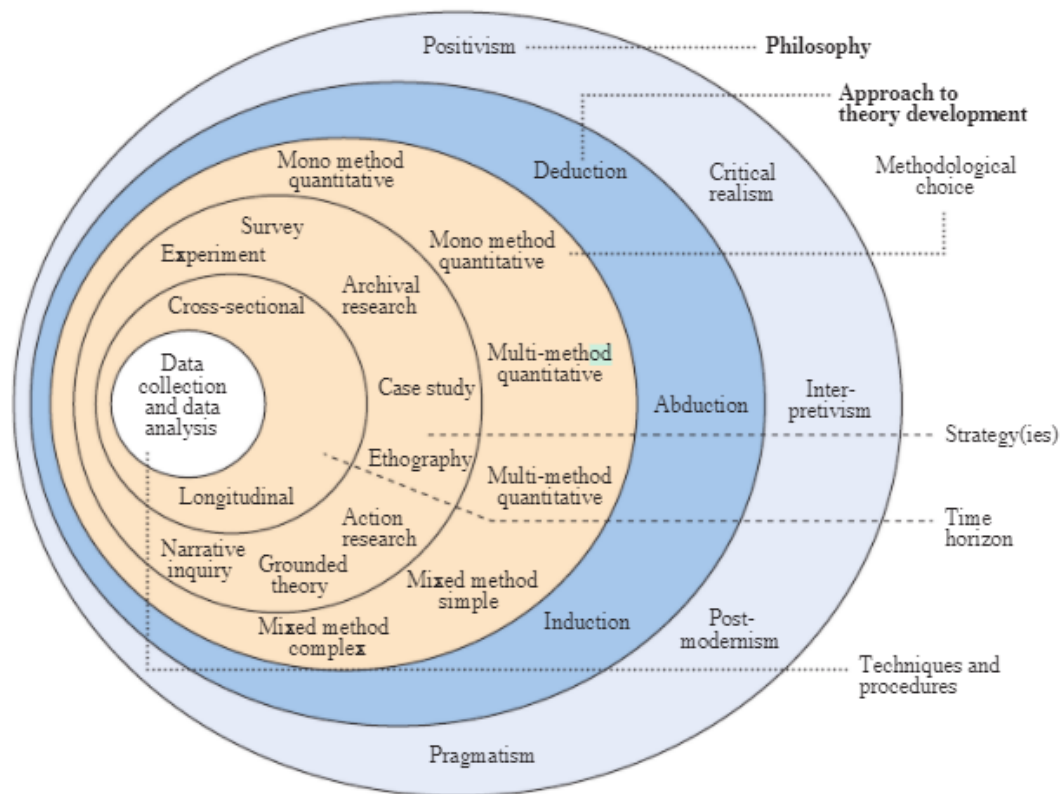


Figure 26: Research Onion (Melnikovas, 2018).

The model proposed in the research onion is made up of six layers. Starting by the selection of the main **research philosophy**, followed by the selection of the **type of approach** adopted for the research, next is the adoption of a type of **method, strategy**, time **horizon** and finally **techniques and procedures**. The research onion is a tool that enables researchers to structure their research and design effective research models by following the layers of the research onion. It is important to mention however that the research onion was originally designed for business studies and its default form isn't adapted for research about the future (Melnikovas, 2018).

3. Research Philosophy

For research to take form, a starting point is critical. The starting point can be the gap in the research or a solution to an identified problem. In both cases, the research methodology is needed to structure and keep the research within its scope. The same way research depends on its research methodology, the research methodology strongly depends on its philosophical theory and from the philosophical theory, the strategies and techniques of the research can be derived.

There are four main research philosophy: positivist, interpretivism, pragmatist, and critical realist. For this research, two philosophy are adopted: critical realist and positivist. In this case, the critical realist is used because the research is based on experimentation, training and is intended to be used in a real-life situation. Failure can result in loss of life. The positivist philosophy is also used due to the training stage of the research, the performance of the algorithm needs to be measured to determine if it's fit for real-life situation used. Positivist is an efficient method to obtain a reliable measuring mechanism.

4. Research Approach

According to the research onion in Figure 26 they are three research approaches: deductive, inductive, and abductive. However, the research philosophy selected for this research narrows the choice to the **abductive** approach. The **abductive** approach is excellent for this research because the aim while using this approach is to “identify structures, connections, contexts and constraints, involves the use of cognitive argumentation” (Melnikovas, 2018). In this research we discuss the structure of classifiers, identify connections between the classifiers and the processing techniques, select databases based on the context they are used into and face constraints with each of the classifiers and preprocessing techniques applied.

5. Research Strategy

The Research onion indicates that they are essentially seven research strategies: experimentation, surveys, case studies, action research, ground theories, ethnography, and archival research. For this research, the best research strategy to adopt is the **Experimentation** strategy since experiments are done to improve the accuracy of the classifiers.

6. Data Collection and Data Analysis

This is the most important stage of the research onion. At this stage, the researcher needs to determine the appropriate techniques and methods to adopt for the data collection and analysis. In this research, the data used is secondary data. It is collected from online resources such as tweets and other social network-related data. Various tools are also used for the preprocessing and analysis of the data. The data is mainly used for the training of the machine learning algorithm.

7. Legal and Ethical Consideration

The proposed research must comply with related legal laws and Ethical issues. The researcher must take the academic guidance and policies seriously and follow the academic code of conduct during the research activities. The research elements, events, and actions regarding theoretical and practical studies, data collection and data analysis methods must comply with legal regulations.

The research approaches, mechanisms, activities, computer, and Internet uses must follow legal requirements and obligations in the United Kingdom (legislation.gov.uk, CMA, 1990). The data collect will be used for the sole purpose of this project and deleted after usage. Only data in English is to be collected and analysed and must comply with DPA (2018) and the GDPR (2018) (legislation.gov.uk).

V. Implementation

According to the objectives of this dissertation, a machine learning-based approach is proposed to detect and categorize offensive slangs and unacceptable language. Furthermore, the approach is used in a real-life situation to evaluate the performance of the proposed approach.

In this section, the pre-processing method used to clean the dataset is first reviewed followed by a detailed description of the categorization algorithm used and finally to simulate a real-life situation a chat app is developed to test the algorithm's performance.

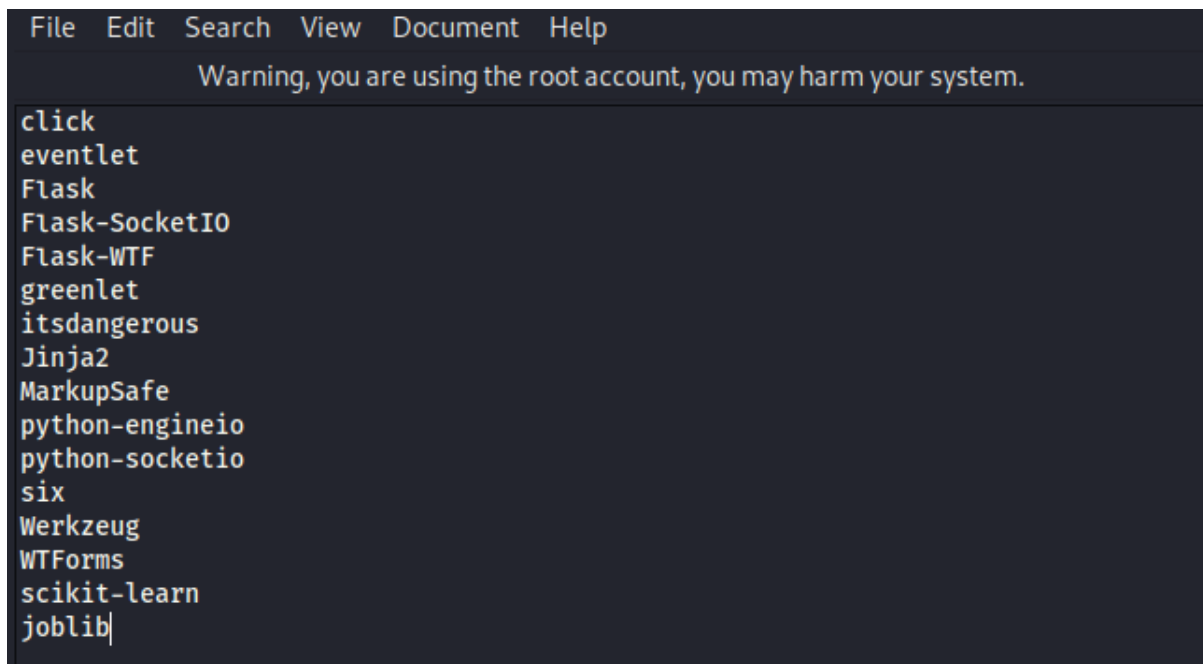
1. Implementation Tools and Prerequisite Setup

1.1.Tools

The implementation is done using “**Kali Linux**” for optimal use of “**python**” and “**PyCharm**”, a Python IDE for professional developers. It is a flexible yet robust python development tool that facilitates coding by using intelligent code completion, on-the-fly error checking, quick-fixes, and many other tools.

1.2.Prerequisite Setup

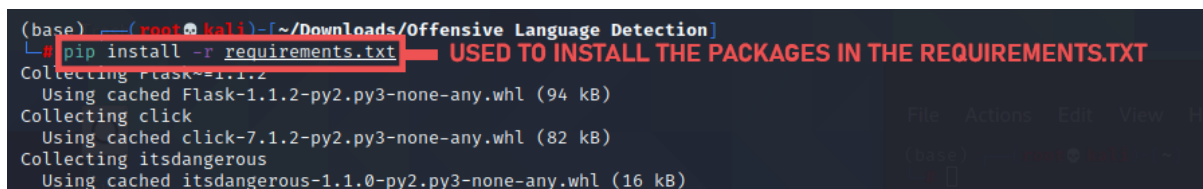
The prerequisite setup involves the installation of **Python 3.6**, **pip** and the following list of **requirements** showed in Figure 27 to run the code without error:



```
File Edit Search View Document Help
Warning, you are using the root account, you may harm your system.
click
eventlet
Flask
Flask-SocketIO
Flask-WTF
greenlet
itsdangerous
Jinja2
MarkupSafe
python-engineio
python-socketio
six
Werkzeug
WTForms
scikit-learn
joblib
```

Figure 27: Requirements for Python code to function properly.

To install the requirements, the command in Figure 28 is used:



```
(base) —(root@kali)~[~/Downloads/Offensive Language Detection]
# pip install -r requirements.txt USED TO INSTALL THE PACKAGES IN THE REQUIREMENTS.TXT
Collecting Flask==1.1.2
  Using cached Flask-1.1.2-py2.py3-none-any.whl (94 kB)
Collecting click
  Using cached click-7.1.2-py2.py3-none-any.whl (82 kB)
Collecting itsdangerous
  Using cached itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
```

Figure 28: Install Packages in Requirements.txt

Once the packages are installed, PyCharm is installed and the rest of the process indicated in section 2 can be followed.

2. Dataset Selection and Pre-Processing

2.1.Dataset Selection

A dataset proposed by (Zhou, 2019) is selected and used to train the algorithm. The dataset is the unification of two other datasets: a “**hate speech and offensive language**” Twitter dataset and a “**Wikipedia**” dataset.

The Twitter dataset contains a column named **class**. In the original dataset, the column was designed to have three values: **0** if it contains hate speech, **1** if it contains offensive language and **2** if it contains neither. For this research, “**class**” was redesigned such if its value is **2** if the text data is classified as “**not offensive**” and all other data is considered as “**offensive**”.

On the other hand, the original Wikipedia dataset was designed with multiple binary columns classified as **toxic** or **threat**. For this research, the dataset is also slightly modified such as any text with toxicity or threat is considered as “**offensive**” while all other text is classified as “**not offensive**”.

2.2.Pre-Processing, Indexing and Feature Extraction

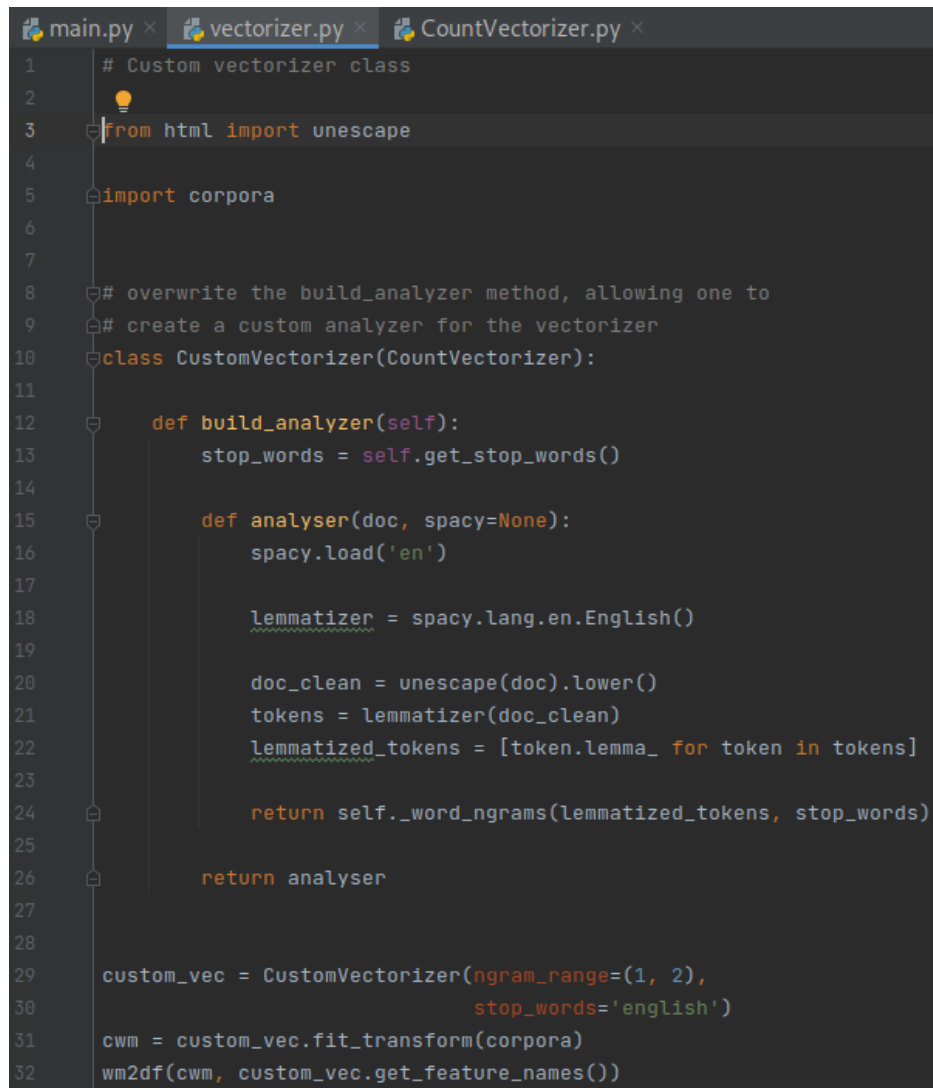
As previously mentioned, the preprocessing stage aims at processing a document's content into a less ambiguous format. A combined method using both a preprocessor (bag-of-words) and a vectorizer (scikit-learn vectorizer) is used for the preprocessing stage. Each of these models improves the preprocessing method by using methods specific to them.

The bag of words model functions by performing three main tasks: **first**, the documents are split into tokens based on a certain pattern **next** weights are assigned to each token based on the frequency at which they occur within the document and **finally**, a document-term matrix is created with each row indicating a document and each column representing a token.

The scikit learn vectorizer present various advantages: it can perform most the tasks performed by the bag of words, it can execute other preprocessing methods and finally, it can apply rules to documents regarding the number and frequency of tokens. They are three main scikit learn vectorizer: The Count Vectorizer, the Hash Vectorizer and the TD-IDF Vectorizer.

The **Count Vectorizer** just like the name suggests counts the number of times a token appears in a document and uses the value as its weight. The **Hash Vectorizer** is designed to be a memory-efficient method, it applies a hashing trick to encode tokens as numerical indexes. However, due to this process, the feature's named cannot be retrieved afterwards. Finally, the **TF-IDF (Term Frequency-Inverse Document Frequency) Vectorizer** is a more efficient method since it assigns weight to tokens not only based on their frequency in the document but also on how recurrent the token is within the entire document.

Although these methods present various advantages, for the preprocessing at this stage a customize vectorizer is used as indicated in Figure 29.

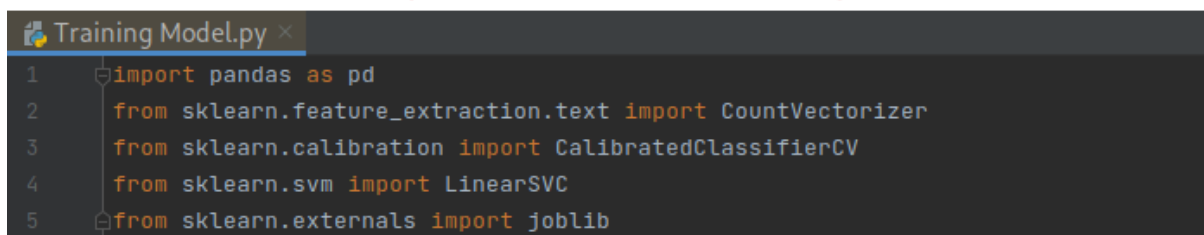


```
1 # Custom vectorizer class
2
3 from html import unescape
4
5 import corpora
6
7
8 # overwrite the build_analyzer method, allowing one to
9 # create a custom analyzer for the vectorizer
10 class CustomVectorizer(CountVectorizer):
11
12     def build_analyzer(self):
13         stop_words = self.get_stop_words()
14
15         def analyser(doc, spacy=None):
16             spacy.load('en')
17
18             lemmatizer = spacy.lang.en.English()
19
20             doc_clean = unescape(doc).lower()
21             tokens = lemmatizer(doc_clean)
22             lemmatized_tokens = [token.lemma_ for token in tokens]
23
24             return self._word_ngrams(lemmatized_tokens, stop_words)
25
26         return analyser
27
28
29 custom_vec = CustomVectorizer(ngram_range=(1, 2),
30                               stop_words='english')
31 cwm = custom_vec.fit_transform(corpora)
32 wm2df(cwm, custom_vec.get_feature_names())
```

Figure 29: Custom vectorizer.

Taking a look at the code, the first section displayed in Figure 30 is used to import the various library necessary to code.

NECESSARY LIBRARY IMPORTS



```
1 import pandas as pd
2 from sklearn.feature_extraction.text import CountVectorizer
3 from sklearn.calibration import CalibratedClassifierCV
4 from sklearn.svm import LinearSVC
5 from sklearn.externals import joblib
```

Figure 30: Library Import

In this portion of the code, a custom analyser is built to first overwrite the `build_analyser` method and next perform the pre-processing phase. The analyser performs three tasks. It first executes **pre-processing**, followed by **tokenization**, and finally **removes stop words** and **extracts n-grams**. Figure 31 gives an overview of the task of each portion of the code.

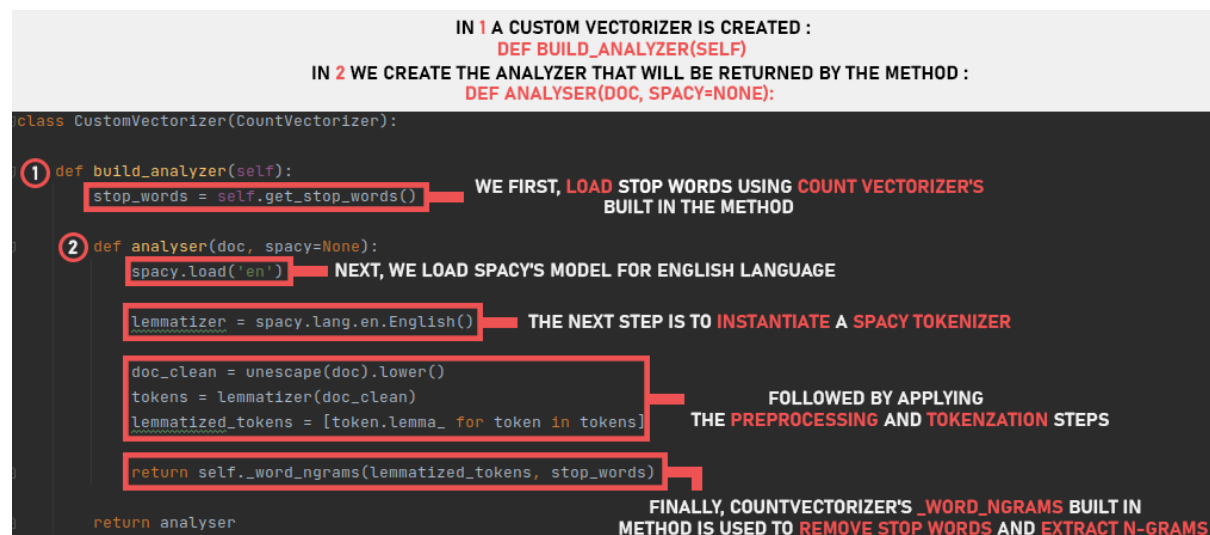
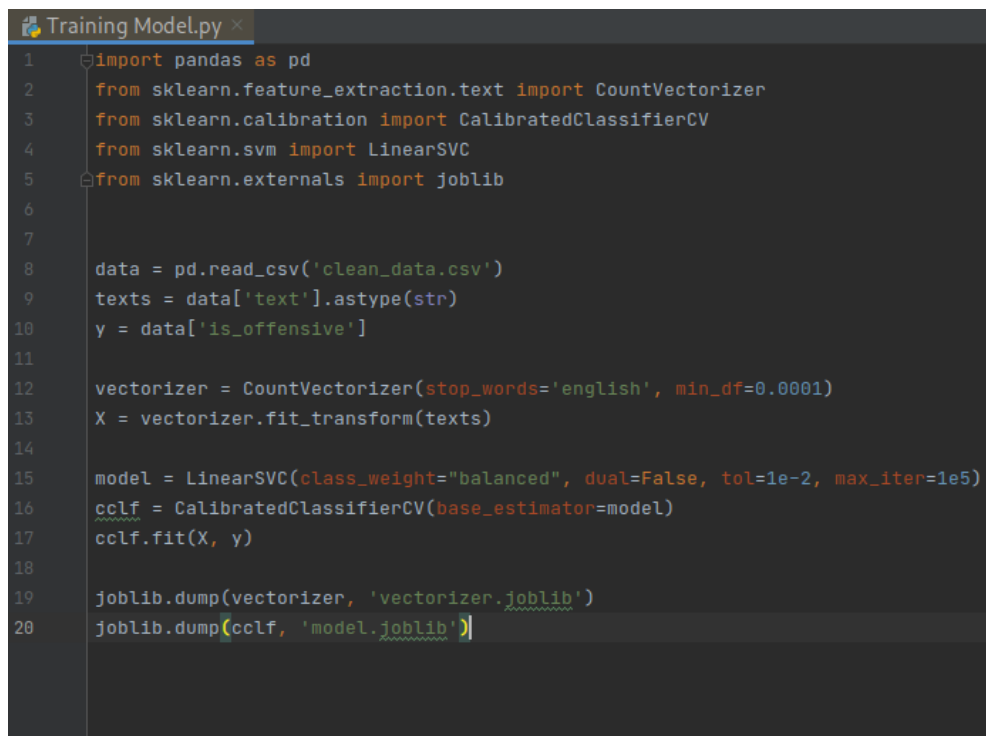


Figure 31: Custom Analyser for the vectorizer.

Once the dataset is processed by the custom vectorizer, the result is a clean and combined dataset that can now be used to train the model.

3. Classifier Identification and Training

In this section, the Linear Support Vector Machine (SVM) algorithm is used to perform the classification. As previously mentioned, Linear SVM is a supervised machine learning model. With the used scikit learn, the Linear SVM is trained to identify offensive and non-offensive words. Due to the abilities of the scikit-learn model to perform classification, regression, clustering, dimensionality reduction, model selection and preprocessing, most of the step needed for the training are automated and called upon using scikit learn functions. Figure 32 gives an overview of the code used to perform the training of the Linear SVM algorithm.



```

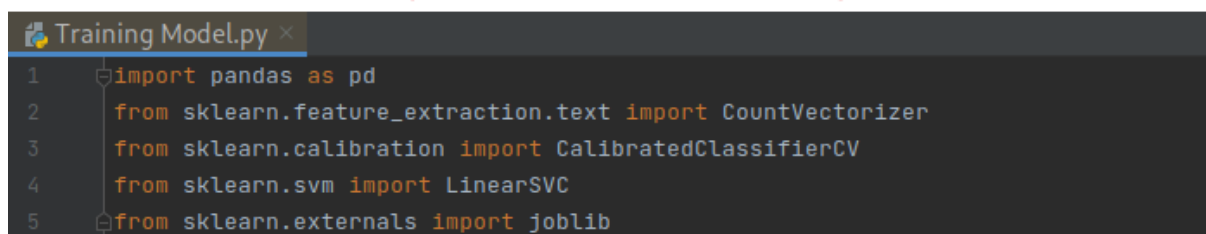
1 import pandas as pd
2 from sklearn.feature_extraction.text import CountVectorizer
3 from sklearn.calibration import CalibratedClassifierCV
4 from sklearn.svm import LinearSVC
5 from sklearn.externals import joblib
6
7
8 data = pd.read_csv('clean_data.csv')
9 texts = data['text'].astype(str)
10 y = data['is_offensive']
11
12 vectorizer = CountVectorizer(stop_words='english', min_df=0.0001)
13 X = vectorizer.fit_transform(texts)
14
15 model = LinearSVC(class_weight="balanced", dual=False, tol=1e-2, max_iter=1e5)
16 cclf = CalibratedClassifierCV(base_estimator=model)
17 cclf.fit(X, y)
18
19 joblib.dump(vectorizer, 'vectorizer.joblib')
20 joblib.dump(cclf, 'model.joblib')

```

Figure 32: SVM training used pre-processed dataset and scikit learn.

Taking a look at the code, the first section displayed in Figure 33 is used to import the various library and other imports necessary to code.

NECESSARY LIBRARY IMPORTS



```

1 import pandas as pd
2 from sklearn.feature_extraction.text import CountVectorizer
3 from sklearn.calibration import CalibratedClassifierCV
4 from sklearn.svm import LinearSVC
5 from sklearn.externals import joblib

```

Figure 33: Library Import 2

As previously mentioned, scikit-learn will be the main element to help with the training. First Pandas library is imported as “**pd**” and used for data analysis using its DataFrame structure. In this case, Panda is used to import the clean dataset saved under the “**CSV**” format.

Next, scikit-learn **CountVectorizer Class** is imported. In this case, it is used to convert all text strings into vectors based on the number of occurrences in the document.

The **CalibratedClassifierCV** is then used as a wrapper to allow the use of the **predict_proba()** method. This method is used to return the probability for each class individually instead of the whole classification process.

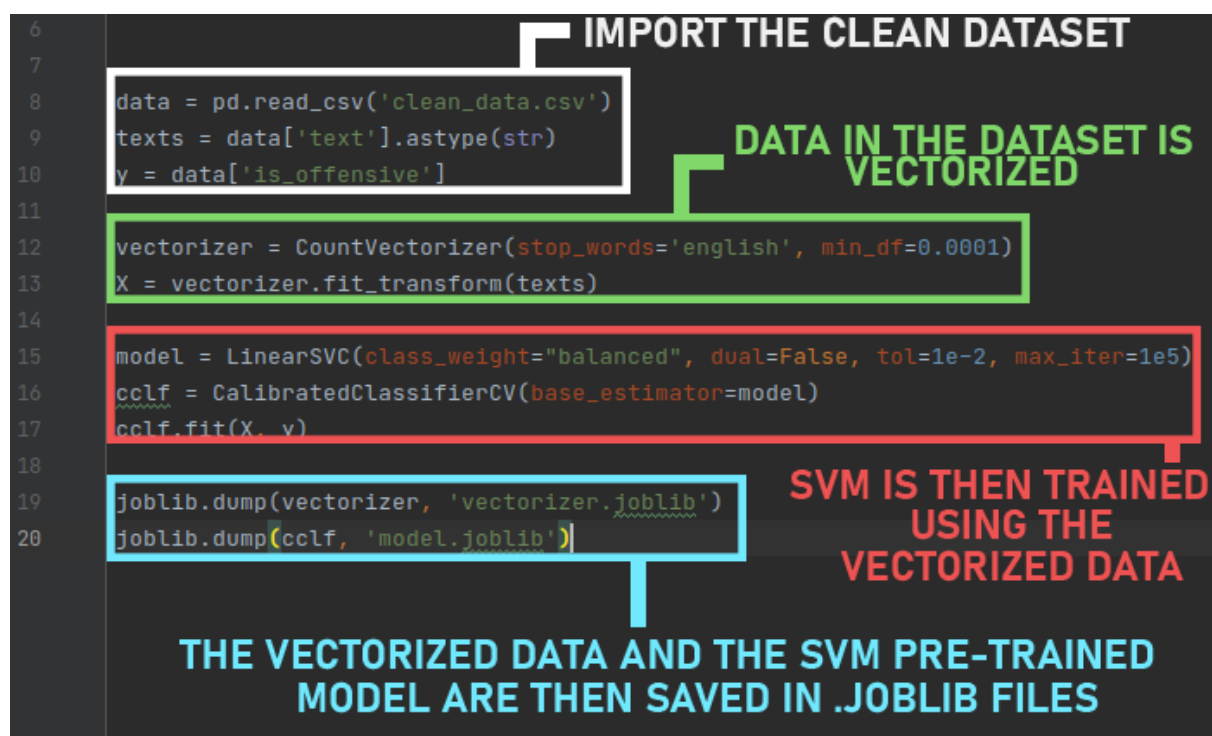
Next, the **Linear SVM model** is imported and implemented by scikit-learn's **LinearSVC Class**.

Finally, **joblib** is also imported and provides a set of tools to facilitate pipelining in python. It is mainly used as a **fast** and **robust** method for large data and allows **specific implementation on "NumPy" arrays**.

In the first section of Figure 34, the **dataset is imported** in the project using the **Pandas DataFrame structure**. It is then **converted** from its **text structure** into a **string format (str)**. Next, using the **CountVectorizer**, the data in the dataset is vectorized following the **rule** previously discussed.

Following the vectorization, the most important part is the **Training Stage** of the Linear SVM is done using scikit-learn's LinearSVC. During this process, the Linear SVM learns about the words considered as offensive thanks to the preprocessing stage where they were labelled and assigned weights.

Finally, the **vectorized data is saved** using the **joblib class** and the **pre-trained Linear SVM model is saved** as well. These files will be used during the evaluation phase to test the performance of the trained SVM.



```
6
7
8 data = pd.read_csv('clean_data.csv')
9 texts = data['text'].astype(str)
10 y = data['is_offensive']
11
12 vectorizer = CountVectorizer(stop_words='english', min_df=0.0001)
13 X = vectorizer.fit_transform(texts)
14
15 model = LinearSVC(class_weight="balanced", dual=False, tol=1e-2, max_iter=1e5)
16 cclf = CalibratedClassifierCV(base_estimator=model)
17 cclf.fit(X, y)
18
19 joblib.dump(vectorizer, 'vectorizer.joblib')
20 joblib.dump(cclf, 'model.joblib')
```

IMPORT THE CLEAN DATASET

DATA IN THE DATASET IS VECTORIZED

SVM IS THEN TRAINED USING THE VECTORIZED DATA

THE VECTORIZED DATA AND THE SVM PRE-TRAINED MODEL ARE THEN SAVED IN .JOBLIB FILES

Figure 34: Vectorization of dataset and Training of Linear SVM.

To summarize, a dataset of combine tweets and Wikipedia bot conversation was first collected and pre-processed to get an unambiguous dataset. The dataset was then used to train a Linear SVM supervised machine learning algorithm and the pre-trained model was saved to be used during the Testing and Performance Evaluation Stage.

4. Design of the Chat Application

The Chat Application designed in this section is a simple web application that allows communication between two users. To design such an application, three elements are essential: Flask, SocketIO and jQuery.

Each of these elements brings valuable abilities for the creation of the application. Flask is a robust framework that provides simplicity, flexibility, and various features. “SocketIO is a cross-browser JavaScript library that abstracts the client application from the actual transport protocol” (Alla, 2018). To simply, SocketIO could be considered as a gate that enables the communication between computers. It can be implemented in “**flask**” using the “**Flask-SocketIO**” extension that gives flask applications access to bi-directional communication. Furthermore, Socket.JS., a JavaScript library is used to support the client-side of the application. In this section, the various classes, HTML template, forms and other elements used to create the application are described.

4.1.Main Interface (Python & HTML)

To design the application, the first task was to design a **LoginForm** that takes as input the **name of the user** and the **name of the room** the user would like to join. To do that, the **LoginForm** is designed such as it automatically loads into a portion of the HTML page. Figure 35 gives an overview of the **LoginForm** designed in **python**.

```

1  from flask_wtf import Form
2  from wtforms.fields import StringField, SubmitField
3  from wtforms.validators import Required
4
5
6  class LoginForm(Form):
7      """Accepts a nickname and a room."""
8      name = StringField('Name', validators=[Required()])
9      room = StringField('Room', validators=[Required()])
10     submit = SubmitField('Enter Chatroom')
11

```

Figure 35: LoginForm Python

The **LoginForm** in Figure 35 is loaded into the form tag : `<form method="POST">` `</form>` in Figure 36.

```

1  <html>
2  <head>
3      <title>Flask-SocketIO-Chat</title>
4  </head>
5  <body>
6      <h1>Flask-SocketIO-Chat</h1>
7      <form method="POST">
8          {{ form.hidden_tag() }}
9          {{ form.name.label }}: {{ form.name() }} {% for error in form.name.errors %}{{ error }}{% endfor %}<br>
10         {{ form.room.label }}: {{ form.room() }} {% for error in form.room.errors %}{{ error }}{% endfor %}<br>
11         {{ form.submit() }}
12     </form>
13 </body>
14 </html>
15

```

Figure 36: Login Form HTML.

Finally, Figure 37 gives an overview of the first interface of the chat application.

Figure 37: Chat Application Login Interface.

4.2.Routes (HTML)

After making the HTML login interface and loading the **LoginForm**, the next step is to determine and set what is to be done when the button “**Enter Chatroom**” is pressed. As displayed in Figure 38, the first change by calling the **LoginForm** “**form=LoginForm()**”.

```
1 from flask import session, redirect, url_for, render_template, request
2 from . import main
3 from .forms import LoginForm
4
5
6 @main.route('/', methods=['GET', 'POST'])
7 def index():
8     """Login form to enter a room."""
9     form = LoginForm()
10
11     if form.validate_on_submit():
12         session['name'] = form.name.data
13         session['room'] = form.room.data
14         return redirect(url_for('.chat'))
15     elif request.method == 'GET':
16         form.name.data = session.get('name', '')
17         form.room.data = session.get('room', '')
18         return render_template('index.html', form=form)
19
20 @main.route('/chat')
21 def chat():
22     """Chat room. The user's name and room must be stored in
23     the session."""
24     name = session.get('name', '')
25     room = session.get('room', '')
26     if name == '' or room == '':
27         return redirect(url_for('.index'))
28     return render_template('chat.html', name=name, room=room)
29
```

NECESSARY IMPORTS

VALIDATE ONLY IF THE FORM IS SUBMITTED WITH NAME AND ROOM AND PROCEED TO THE .CHAT PAGE ELSE REQUEST FOR THE MISSING PARAMETER (ROOM OR NAME) AND STAY ON INDEX.HTML

THE NAME OF BOTH THE USER AND THE ROOM ARE TO BE STORED IN THE SESSION AND PASSED DOWN TO THE NEXT PAGE

Figure 38: Routes.

4.3.Second Interface (HTML)

Once the user inserts a username and room name, the next interface is loaded. On this interface, users can exchange messages. Figure 39 gives an overview of the HTML code used to develop the chat interface of the web application.

```
1 <html>
2 <head>
3 <title>Flask-SocketIO-Chat: {{ room }}</title>
4 <script type="text/javascript" src="//code.jquery.com/jquery-1.4.2.min.js"></script>
5 <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/socket.io/1.3.6/socket.io.min.js"></script>
6 <script type="text/javascript" charset="utf-8">
7   var socket;
8   $(document).ready(function(){
9     socket = io.connect('http://' + document.domain + ':' + location.port + '/chat');
10    socket.on('connect', function() {
11      socket.emit('joined', {});
12    });
13    socket.on('status', function(data) {
14      $('#chat').val($('#chat').val() + '< ' + data.msg + '>\n');
15      $('#chat').scrollTop($('#chat')[0].scrollHeight);
16    });
17    socket.on('message', function(data) {
18      $('#chat').val($('#chat').val() + data.msg + '\n');
19      $('#chat').scrollTop($('#chat')[0].scrollHeight);
20    });
21    $('#text').keypress(function(e) {
22      var code = e.keyCode || e.which;
23      if (code == 13) {
24        text = $('#text').val();
25        $('#text').val('');
26        socket.emit('text', {msg: text});
27      }
28    });
29  });
30  function leave_room() {
31    socket.emit('left', {}, function() {
32      socket.disconnect();
33
34      // go back to the login page
35      window.location.href = "{{ url_for('main.index') }}";
36    });
37  }
38 </script>
39 </head>
40 <body>
41 <h1>Flask-SocketIO-Chat: {{ room }}</h1>
42 <textarea id="chat" cols="80" rows="20"></textarea><br><br>
43 <input id="text" size="80" placeholder="Enter your message here"><br><br>
44 <a href="#" onclick="leave_room();">Leave this room</a>
45 </body>
46 </html>
```

Figure 39:Chatting Interface.

Figure 40 shows the chat interface.

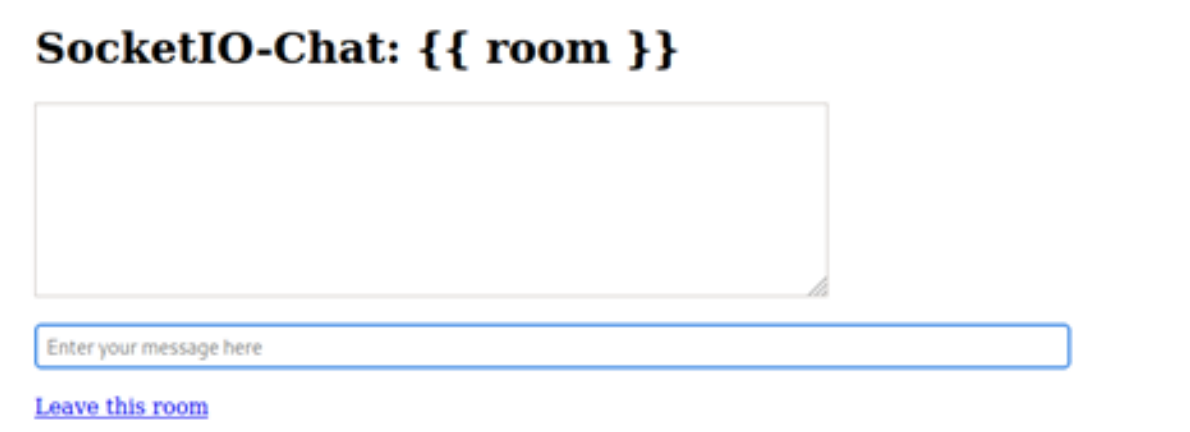


Figure 40: Main Interface (Visual Representation).

4.4.Events (Python)

In this section of the code, **three actions are set**. First what happens when a user **joins the chatroom**, next **what happens and how user emits messages in the chatroom** and finally what happens when a user **leaves the chatroom**. In this section, two actions are reviewed: when a **user joins** set in Figure 41 **and/or leaves the chatroom** set in Figure 42.



Figure 41: Code for a user joining the chatroom.

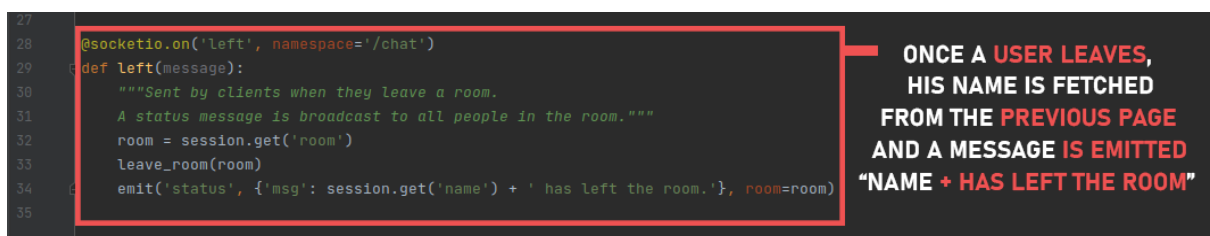


Figure 42: Code for a user leaving the chatroom.

Figure 43 gives an overview of the message emitted when a user joins and leaves a chatroom.

SocketIO-Chat: cx



Figure 43: User joining and leaving a "cx" chatroom.

4.5.Blueprint (Python)

In this case, the Blueprint object is used to define an application function without requiring an application object ahead of time. Simply put, Blueprint is used to defer the need for an application. Figure 44 shows how the Blueprint object is used in this case.

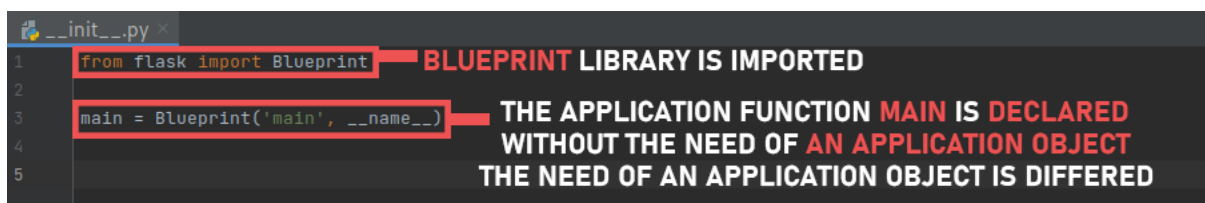


Figure 44: Blueprint used to declare "Main".

4.6.Create the Application (Python)

Once the various interfaces are designed and other conditions are set, the application is created. To do that, a variable named **app** is created by wrapping **Flask** around **__name__**. Next, encryption is enabled by declaring a **SECRET_KEY** and the application declared using blueprint is called and declared making this an application object (main) (Alla, 2018).

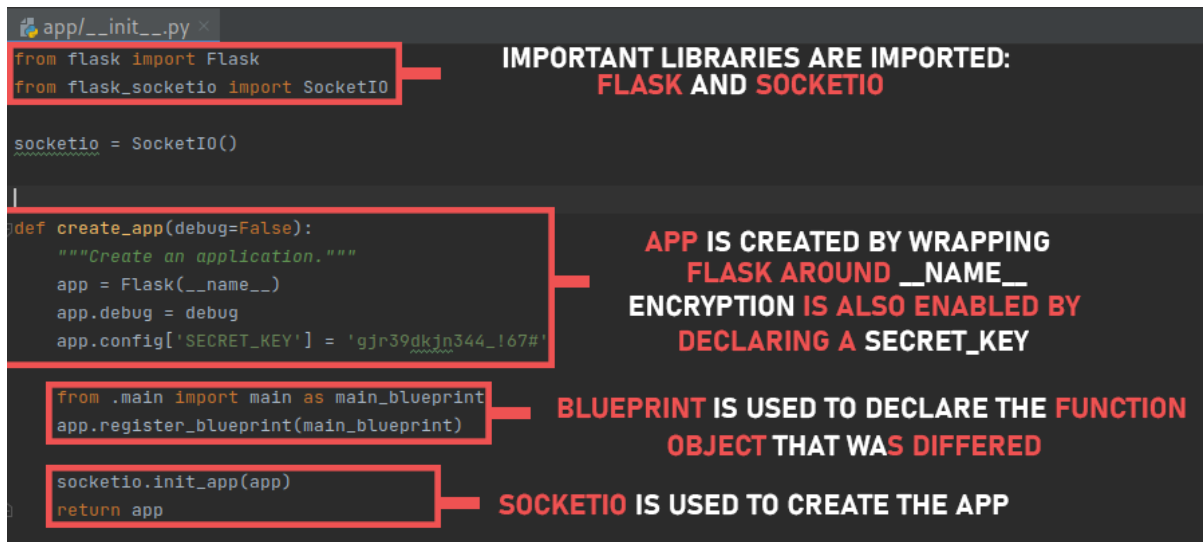


Figure 45: Code used to create the Application.

4.7.Chat Application (Python)

Chat.py is used to regroup all the other elements declared. It can be considered as a setup. As displayed in Figure 46, the variable app previously created is called and the application is such as it listens to all addresses.



Figure 46: Chat Application Launcher.

4.8.Final Chat Application

Once the chat application is created, after running it, the first interface is displayed, and the user is directed to the second interface after inserting his name and the name of the room just as indicated in Figure 47:

SocketIO-Chat

Name:

Room:

USER **JOHN** JOINS THE **DISCUSSION** CHATROOM AND IS DIRECTED TO INTERFACE 2

SocketIO-Chat: **Discussion**

<John has entered the room.>

Enter your message here

[Leave this room](#)

THE NAME OF THE ROOM IS ADDED TO THE TOP AND THE WHOLE CHATROOM IS NOTIFIED THAT JOHN HAS ENTERED THE ROOM

Figure 47: User John logging into the chatroom.

The second user also accesses the room as indicated in Figure 48.

SocketIO-Chat

Name:

Room:

USER **WARREN** JOINS THE **DISCUSSION** CHATROOM AND IS DIRECTED TO INTERFACE 2

SocketIO-Chat: **Discussion**

<Warren has entered the room.>

Enter your message here

[Leave this room](#)

THE NAME OF THE ROOM IS ADDED TO THE TOP AND THE WHOLE CHATROOM IS NOTIFIED THAT WARREN HAS ENTERED THE ROOM

Figure 48: User Warren logging into the chatroom.

Once the second user accesses the chatroom, the first user is also notified and can initiate a conversation. Figure 49 shows how the chat application works.

SocketIO-Chat: Discussion

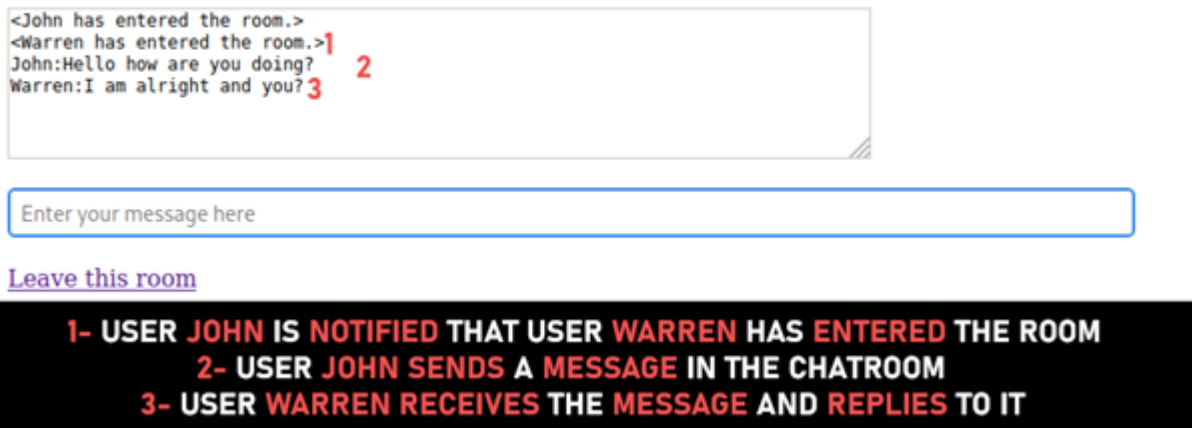


Figure 49: Conversation between two users in the chat app.

Once the chat application is working, the next step is to initiate the process that allows the pre-trained Linear SVM model to identify offensive language within the chat application the next section reviews the method used and shows how the model embedded in the chat application works.

4.9. Demonstration and Evaluation of Offensive Language in Chat Application

With the chat application working, the next step is to integrate our trained model in it. It is done as displayed in Figure 50.



Figure 50: Embedding the pre-trained SVM model in the chat application.

In Figure 50, the vectorizer and model saved in Figure 34 are loaded to be used by the chat app. The vectorizer is used to vectorize chat messages sent in the chatroom. The vectorized text is then passed to the Linear SVM pre-trained model to determine if a text is offensive or not and a Json result is sent as a response.

Finally, the application is set to run on host 0.0.0.0 and on port 8081 to allow it to listen to all addresses. One more change is necessary for the application to work as intended. To the “Event” file modified in Figure 41 and Figure 42, is one function. This function determines how a message sent from a user is displayed in the chatroom. Figure 51 gives an overview of the standard method used.



Figure 51: Default function used to send a message in the chatroom.

However, because the model is added to the chat application, this section of the code is changed to enable the model to function properly.

```
@socketio.on('text', namespace='/chat')
def text(message):
    """Sent by a client when the user entered a new message.
    The message is sent to all people in the room."""
    room = session.get('room')
    # requests module used to send a HTTP request
    import requests
    # request the flask app using the message as the input text
    response = requests.post(url='http://localhost:8081', data={'text': message['msg']}).json()['result']
    # append the response to the message in brackets
    emit('message', {'msg': session.get('name') + ':' + message['msg'] + ' [' + response + ']', 'room': room})
```

Figure 52: Edited function to send a message in the chatroom.

In Figure 52, the function is modified such as text messages sent in the chatroom are sent to the Flask application through an HTTP request. The response to that request is appended and displayed in the chatroom. As a result of this change, the model is fully functional within the chat application and a conversation between two users goes as indicated in Figure 53.

SocketIO-Chat

```
<Oslo has entered the room.>
<John has entered the room.>
Oslo:hello how are you?[Not Offensive]
John:I am doing great my friend and you?[Not Offensive]
Oslo:I am okay too thank you.[Not Offensive]
John:I hate you![Offensive]
Oslo:get the fuck out of here[Offensive]
John:I can't talk to you[Not Offensive]
Oslo:i will beat the shit out of you![Offensive]
```

[Leave this room](#)

Figure 53: Chat with Text Classification Linear SVM embedded.

VI. Evaluation and Recommendation

At the implementation stage, a dataset was first pre-processed through the use of **tokenization**, **remove stop words** and **extracts n-grams** method. The clean dataset from the pre-processing stage is then used to train the SVM algorithm. To that end, a vectorizer is built to process the data and feed the vectorized data to the SVM model for training and testing. However, before the vectorized data is used for training by the algorithm, it is first divided into 70% and 30%.

The 70% is used for the training of the algorithm while the 30% is used for the testing stage. Once the algorithm is trained, the 30% of the vectorized data is used to test and improve on the reaction of the algorithm when faced with new, unknown data. Once the results are acceptable, the vectorizer and the trained model are saved as Joblib.

Following the training and testing of the algorithm, a web application is built to simulate a chatting environment. At this stage, the aim is to demonstrate the effectiveness of the model as well as indicate the impact this model has on the way conversations will made on social media.

The web application is built using SocketIO and Flask and is embedded with the pre-trained model such as chat messages sent in the chatroom are feed to the vectorizer for pre-processing and then to the algorithm for classification. At the end of this process, the algorithm is capable of classifying offensive and not offensive messages.

This result gives an overview of the impact machine learning methods can have on the use of abusive language on social media. By implementing the model into a chat environment, it is possible to identify offensive messages and decide on an appropriate reaction to them. It can vary from discarding them or stop users from sending those messages and many more. The action to perform once the detection is done is out of the scope of this project.

However, a recommendation on the ideal form of this model will be a method that allows the implementation of the text classification model into access points hub as a web application. Allowing users to set the level of profanity allowed to be displayed. Such model would provide users with the flexibility and total control over to decide what level is acceptable or not and what features are activated or not (general classification, chat classification and many more).

VII. Conclusion and Future Work

In this research, an in-depth literature review was done on various text classification methods used for preprocessing, training, feature extraction, data disambiguation and more. Furthermore, an analysis of popular datasets and approaches used to achieve the text classification goal was also discussed. After discussing these methods, techniques and approaches, a text classification process was proposed, it encompasses existing stages but also provides a more detailed step-to-step process. The proposed process was further discussed, and approaches used by previous research at each stage was discussed. Although most of the approaches to TC achieve their goal at text classification, it was identified that the focus has on comments and posts.

Following this observation, an attempt to text classification in the context of messaging and “real-life situation” was done. To achieve that goal, a hate speech and offensive language dataset were first preprocessed using two methods a bag-of-words preprocessor and a scikit-learn vectorizer. Resulting in a clean dataset that can be used to train the classifier. For this research, a Linear SVM classifier is used and with the dataset collected from the previous stage, the Linear SVM is trained using the Scikit-Learn functions. Once trained, the vectorizer and trained SVM are saved and embedded in a chat application. The vectorizer will be used by the chat application to vectorize chat conversations and transfer the vectorized chat to the pre-trained model which determines if the text is offensive or not. The chat application, in this case, is a web application developed with the help of Socket-IO a JavaScript library and the Flask-SocketIO extension. Once the model is embedded in the web application, the experimental stage begins.

After experimenting on the application, it was identified that to some extent, the pre-trained model was able to detect “offensive language” in a message conversation with an accuracy of 95% and a precision of 86.1%. Although the result is promising, it is important to mention that the model has difficulties with a less common variant of abusive languages and hence makes mistakes.

To further improve on this research, an attempt to use a deep learning method for the text classification should be examined with an emphasizes on uncommon abusive language. Also, future work should aim at the implementation of classification models into live models to solve other social problems.

VIII. References

- Aggarwal C.C., Z. C., 2012. A Survey of Text Classification Algorithms. In: *Mining Text Data*. Springer: Boston, pp. 163-222.
- Ahluwalia, R. et al., 2018. *Detecting Hate Speech Against Women in English Tweets*. s.l., s.n.
- Albitar, S., Espinasse, B. & Fournier, S., 2012. *Towards a Supervised Rocchio-based Semantic Classification of Web Pages*. San Sebastian, Spain, s.n.
- Allahyari, M. et al., 2017. A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques. *ArXiv*.
- Alla, S., 2018. *Building your first Chat Application using Flask in 7 minutes*. [Online] Available at: <https://codeburst.io/building-your-first-chat-application-using-flask-in-7-minutes-f98de4adfa5d> [Accessed 16 december 2020].
- Al-Makhadmeh, Z. & Tolba, A., 2020. Automatic hate speech detection using killer natural language processing optimizing ensemble deep learning approach. *Computing*, Volume 102, pp. 501-522.
- Al-Shammari, E. & Lin, J., 2008. *A Novel Arabic Lemmatization Algorithm*. New York, s.n.
- amungkas, E. W., Cignarella, A. T., Basile, V. & Patti, V., 2018. *Automatic Identification of Misogyny in English and Italian Tweets at EVALITA 2018 with a Multilingual Hate Lexicon*. s.l., s.n.
- Aramaki, E. & Miyo, K., 2006. *Patient Status Classification by using Rule based Sentence Extraction and BM 25-kNN based Classifier*. s.l., s.n.
- Attia, M., Ziriky, A. & Diab, M. T., 2016. *The Power of Language Music: Arabic Lemmatization through Patterns*. Osaka, Japan,, s.n.
- Bakarov, A., 2018. *Vector Space Models for Automatic Misogyny Identification (Short Paper)*. s.l., s.n.
- Bansal, H., Shrivastava, G., Nhu, N. & Stanciu, L., 2018. *Social Network Analytics for Contemporary Business*. Pennsylvania, USA: IGI Global: Hershey.
- Basile, A. & Rubagotti, C., 2018. *A performant, cross-lingual misogyny detection system..* s.l., s.n.
- Basile, P., Caputo, A. & Semeraro, G., 2014. *An Enhanced Lesk Word Sense Disambiguation Algorithm through a Distributional Semantic Model*. Dublin, Ireland, s.n.
- Bateman, J. & Zock, M., 2012. Natural Language Generation. *The Oxford Handbook of Computational Linguistics*, pp. 285-304.
- Bauer, E. & Kohavi, R., 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, Volume 36, pp. 105-139.
- Bengio, Y., Simard, P. & Frasconi, P., 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2), pp. 157-166.
- Benikova, D., Wojatzki, M. & Zesch, T., 2018. *What Does This Imply? Examining the Impact of Implicitness on the Perception of Hate Speech*. s.l., s.n.

- Bitext, 2018. *What is the difference between stemming and lemmatization?*. [Online]
Available at: <https://blog.bitext.com/what-is-the-difference-between-stemming-and-lemmatization/>
[Accessed 16 December 2020].
- Blei, D. M., Ng, A. Y. & Jordan, M. I., 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3, Volume 3, pp. 993-1022.
- Bloehdorn, S. & Hotho, A., Knowledge Discovery on the Web. *Boosting for text classification with semantic features*. Berlin/Heidelberg, s.n.
- Boser, B., Guyon, I. & Vapnik, V., 1992. *A training algorithm for optimal margin classifiers*. Pittsburgh, s.n.
- Boudchiche, M. et al., 2017. AlKhalil Morpho Sys 2: A robust Arabic morpho-syntactic analyzer. *Journal of King Saud University - Computer and Information Sciences*, 29(2), pp. 141-146.
- Bovi, C. D., Telesca, L. & Roberto, N., 2015. Large-Scale Information Extraction from Textual Definitions through Deep Syntactic and Semantic Analysis. In: *Transactions of the Association for Computational Linguistics, Volume 3*. s.l.:s.n., p. 529–543.
- Brants, T., Chen, F. & Tsochantaridis, I., 2002. *Topic-based document segmentation with probabilistic latent semantic analysis*. McLean, Virginia, s.n.
- Breiman, L., 1996. Bagging Predictors. *Machine Learning*, Volume 24, pp. 123-140.
- Breiman, L., 2001. Random Forests. *Machine Learning*, Volume 45, p. 5–32.
- Brin, S. & Page, L., 1998. The anatomy of a large-scale hypertextual Web search engine. In: *Computer Networks and ISDN Systems*. s.l.:Elsevier, pp. 107-117.
- Burnap, P. & Williams, M., 2016. Us and them: identifying cyber hate on Twitter across multiple protected characteristics. *EPJ Data Science*, 5(11).
- Buscaldi & Davide, 2018. *Tweetaneuse @ AMI EVALITA2018: Character-based Models for the Automatic Misogyny Identification Task (Short Paper)*. s.l., s.n.
- Camacho-Collados, J., Pilevar, M. T. & Navigli, R., 2016. NASARI: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. In: *Artificial Intelligence*. s.l.:s.n., pp. 36-64.
- Canós, J. S., 2018. *Misogyny identification through SVM*. s.l., s.n.
- Chihli Hung, S.-J. C., 2016. Word sense disambiguation based sentiment lexicons for sentiment classification. In: *Knowledge-Based Systems*. s.l.: Elsevier, pp. 224-232.
- Community, H. E. S., 2019. *What is parsing in NLP?*. [Online]
Available at: <https://forum.huawei.com/enterprise/en/what-is-parsing-in-nlp/thread/>
[Accessed 19 December 2020].
- Dalal, M. K. & Zaveri, M. A., 2011. Automatic Text Classification: A Technical Review. *International Journal of Computer Applications (0975 – 8887)*, 28(2), pp. 37-40.
- Davidson, T., Warmesley, D. & Macy, M., 2017. *Automated Hate Speech Detection and the Problem of Offensive Language*. s.l., s.n.

De Mántaras, R. L., 1991. A Distance-Based Attribute Selection Measure for Decision Tree Induction. *Machine Learning*, Volume 6, p. 81–92.

Deerwester, S., 1988. *Improving Information Retrieval with Latent Semantic Indexing*. s.l., s.n., pp. 391-407.

Del Vigna, F., Cimino, A. & Dell Orletta, F., 2017. *Hate me, hate me not: Hate speech detection on Facebook*. Venice, s.n.

Djuric, N. et al., 2015. *Hate Speech Detection with Comment Embeddings*. s.l., s.n.

Dootio, M. A. & Wagan, A. I., 2017. AUTOMATIC STEMMING AND LEMMATIZATION PROCESS FOR SINDHI TEXT. Volume 6, pp. 19-28.

Dootio, M. A. & Wagan, A. I., 2019. Syntactic parsing and supervised analysis of Sindhi text. *Journal of King Saud University - Computer and Information Sciences*, 31(1), pp. 105-112.

Dr. Garbade, M. J., 2018. *A Simple Introduction to Natural Language Processing*. [Online] Available at: <https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32> [Accessed 16 December 2020].

Farzi, R. & Bolandi, V., 2016. Estimation of organic facies using ensemble methods in comparison with conventional intelligent approaches: a case study of the South Pars Gas Field, Persian Gulf, Iran. *Modeling Earth Systems and Environment*, Volume 2, p. 105.

Flejter, D., Wieloch, K. & Abramowicz, W., 2007. Unsupervised Methods of Topical Text Segmentation for Polish. In: *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing*. Prague: Association for Computational Linguistics, p. 51–58.

Founta, A.-M. et al., 2018. *Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior*. s.l., s.n.

Frank, E. & Bouckaert, R., 2006. *Naive bayes for text classification with unbalanced classes*. Berlin/Heidelberg, s.n.

Freihat, A. A., Abbas, M., Bella, G. & Giunchiglia, F., 2018. *Towards an Optimal Solution to Lemmatization in Arabic*. Dubai, United Arab Emirates, s.n.

Frenda, S., Ghanem, B. & Montes-y-Gómez, M., 2018. *Exploration of Misogyny in Spanish and English tweets*. s.l., s.n.

Frenda, S., Ghanem, B., Montes-y-Gómez, M. & Villasenor-Pineda, L., 2018. *Automatic Expansion of Lexicons for Multilingual Misogyny Detection*. s.l., s.n.

Freund, Y., 1992. *An improved boosting algorithm and its implications on learning complexity*. Pittsburgh, s.n.

Freund, Y. et al., 1995. *Efficient algorithms for learning to play repeated games against computationally bounded adversaries*. Milwaukee, s.n.

G., G., 2014. Soft Biometrics from Face Images Using Support Vector Machines. In: *Support Vector Machines Applications*. Cham: Springer, pp. 269-302.

Gao, Y. et al., 2010. *Sentiment classification for stock news*. Maribor, s.n.

- Geurts, P., 2000. *Some enhancements of decision tree bagging*. Berlin/Heidelberg, Germany, s.n.
- Giovanelli, C. et al., 2017. *Towards an aggregator that exploits big data to bid on frequency containment reserve market*. Beijing, China, s.n.
- Goddard, C. & Schalley, A. C., 2010. Semantic Analysis. In: *Handbook of Natural Language Processing*. s.l.:CRC Press, Taylor & Francis, pp. 92-120.
- Goenaga, I. et al., 2018. *Automatic Misogyny Identification Using Neural Networks*. s.l., s.n.
- Guerin, A., 2016. *Using Demographic Variables and In-College Attributes to Predict Course-Level Retention for Community College Spanish Students*, Scottsdale: s.n.
- Gupta, D., Yadav, R. K. & Sajan, N., 2011. *A novel corpus-based stemming algorithm using co-occurrence statistics*. s.l., s.n., pp. 1-8.
- Han, E. & Karypis, G., 2000. *Centroid-based document classification: Analysis and experimental results*. Berlin/Heidelberg, s.n.
- Han, E., Karypis, G. & Kumar, V., 2001. *Text categorization using weight adjusted k-nearest neighbor classification*. Berlin/Heidelberg, s.n., pp. 53-65.
- Harish, B. S., Guru, D. S. & Manjunath, S., 2010. Representation and Classification of Text Documents: A Brief Review. *IJCA, Special Issue on "Recent Trends in Image Processing and Pattern Recognition"*, pp. 110-119.
- Hou, S. & Lu, R., 2020. Knowledge-guided unsupervised rhetorical parsing for text. *Information Systems*, Volume 94.
- Huang, K., 2015. *Unconstrained Smartphone Sensing and Empirical Study for Sleep Monitoring and self-management*, Lowell: s.n.
- Huang, X. et al., 2003. Applying Machine Learning to Text Segmentation for Information Retrieval. *Information Retrieval Journal*, Volume 6, p. 333–362.
- Irny, S. & Rose, A., 2005. Designing a Strategic Information Systems Planning. *Issues in Information System*, VI(1).
- Jaderberg, M., Simonyan, K., Vedaldi, A. & Zisserman, A., 2015. Reading text in the wild with convolutional neural. *International Journal of Computer Vision*, Volume 116, pp. 1-20.
- Jain, A., Kulkarni, G. & Shah, V., 2018. Natural Language Processing. *International Journal of Computer Sciences and Engineering*, Volume 6, pp. :161-167.
- Jasim, D. S., 2016. *Data Mining Approach and Its Application to Dresses Sales Recommendation*, Malaysia: Research Gate.
- Johnson, D. E., Oles, F. J., Zhang, T. & Goetz, T., 2002. A decision-tree-based symbolic rule induction system for text categorization. *IBM Systems Journal*, 41(3), pp. 428 - 437.
- Juan, A. & Vidal, E., 2002. On the use of Bernoulli mixture models for text classification.. *Pattern Recognition*, 35(12), pp. 2705-2710.
- Karamizadeh, S. et al., 2014. *Advantage and drawback of support vector machine functionality*. Langkawi, Malaysia, s.n.

- Kim, S.-B., Han, K.-S., Rim, H.-C. & Myaeng, S. H., 2006. Some Effective Techniques for Naive Bayes Text Classification. *IEEE Transactions on Knowledge and Data Engineering*, 18(11), pp. 1457-1466.
- Kowsari, K. et al., 2017. *HDLTex: Hierarchical Deep Learning for Text Classification*. Cancun, s.n.
- Kowsari, K. et al., 2019. Text Classification Algorithms: A Survey. *Information*, Volume 150.
- Krovetz, R., 2000. Viewing morphology as an inference process. *Artificial Intelligence*, 118(1-2), pp. 277-294.
- Kumawat, D., 2019. *7 Natural Language Processing Techniques for Extracting Information*. [Online] Available at: <https://www.analyticssteps.com/blogs/7-natural-language-processing-techniques-extracting-information> [Accessed 19 December 2020].
- Lai, S., Xu, L., Liu, K. & Zhao, J., 2015. *Recurrent Convolutional Neural Networks for Text Classification*. Austin, TX, USA, s.n.
- Leslie, C. et al., 2004. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4), p. 467–476.
- Leslie, C., Eskin, E. & Noble, W., 2002. The spectrum kernel: a string kernel for SVM protein classification. *Pacific Symposium on Biocomputing*, pp. 564-575.
- Lewis, D. & Ringuette, M., 1994. *A comparison of two learning algorithms for text categorization*. Las Vegas, s.n.
- Lin, Y., 2002. Support Vector Machines and the Bayes Rule in Classification. *Data Mining and Knowledge Discovery*, Volume 6, p. 259–275.
- Liu, C., Wang, Y. & Zheng, F., 2006. *Automatic Text Summarization for Dialogue Style*. Weihai, s.n.
- Liu, H., Chiroma, F. & Haig, E., 2018. *Identification and Classification of Misogynous Tweets Using Multi-classifier Fusion*. s.l., s.n.
- Liu, Y., Loh, H. & Sun, A., 2009. Imbalanced text classification: A term weighting approach. *Expert Systems with Applications*, 36(1), pp. 690-701.
- Madukwe, K. J., Gao, X. & Xue, B., 2020. *In Data We Trust: A Critical Analysis of Hate Speech Detection Datasets*. s.l., s.n.
- Mahender, C. N. & Korde, V., 2012. Text classification and classifiers: a survey. *International Journal of Artificial Intelligence & Applications (IJAI)*, 3(2), pp. 85-99.
- Maillo, J. R. S. T. I. & H. F., 2017. kNN-IS: An Iterative Spark-based design of the k-Nearest Neighbors classifier for big data. *Knowledge-Based Systems*, Volume 117, pp. 3-15.
- Malhotra, S. & Godayal, D., 2018. *FreeCodeCamp*. [Online] Available at: <https://www.freecodecamp.org/news/an-introduction-to-part-of-speech-tagging-and-the-hidden-markov-model-953d45338f24/> [Accessed 18 December 2020].
- Malik, U., n.d. *Text Classification with Python and Scikit-Learn*. [Online] Available at: <https://stackabuse.com/text-classification-with-python-and-scikit-learn/> [Accessed 04 January 2021].

- Maron, O. & Lozano-Pérez, T., 1997. *A framework for multiple-instance learning*. s.l., s.n.
- Mehta, M., Agrawal, R. & Rissanen, J., 1996. *SLIQ: A fast scalable classifier for data mining*. Berlin, Heidelberg, s.n.
- Melnikovas, A., 2018. Towards an Explicit Research Methodology: Adapting Research Onion Model for Futures Studies. *Journal of Futures Studies*, 23(2), p. 29–44.
- Melucci, M., 2009. Vector-Space Model. In: *Encyclopedia of Database Systems*. Boston: Springer.
- MonkeyLearn, n.d. *Text Classification*. [Online]
Available at: <https://monkeylearn.com/text-classification/>
[Accessed 16 December 2020].
- Navigli, R., 2009. Word Sense Disambiguation: A Survey. *ACM Computin Surveys*, 41(2), pp. 10:2-10:69.
- Nidhi & Gupta, V., 2011. Recent trends in text classification techniques. *International Journal of Computer Applications*, 35(6), pp. 45-51.
- Nina-Alcocer, V., 2018. *Automatic Misogyny Identification in Spanish*. s.l., s.n.
- Nobata, C. et al., 2016. *Abusive Language Detection in Online User Content*. s.l., s.n.
- Osman, D. J. & Yearwood, J., 2007. *Opinion Search in Web Logs*. Ballarat, s.n.
- Pajarskaite, G., Gričiute, V. & Raskinis, G., 2004. Designing HMM-Based Part-of-Speech Tagger for Lithuanian Language. *Informatica*, Volume 15, pp. 231-242.
- Pak, I. & Teh, P. L., 2018. Text Segmentation Techniques: A Critical Review. In: *Innovative Computing, Optimization and Its Applications*. s.l.:Springer, pp. 167-181.
- Pamungkas, E. W., Basile, V. & Patti, V., 2020. Misogyny Detection in Twitter: a Multilingual and Cross-Domain. *Information Processing and Management*, 57(6).
- Pamungkas, E. W., Cignarella, A. T., Basile, V. & Patti, V., 2018. *14-ExLab@UniTo for AMI at IberEval2018: Exploiting Lexical Knowledge for Detecting Misogyny in English and Spanish Tweets*. s.l., s.n.
- Partalas, I. et al., 2015. *LSHTC: A benchmark for large-scale text classification*, s.l.: arXiv.
- Pasha, A., Elbadrashiny, M., Diab, M. & Elkholy, A., 2014. *MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic*. s.l., s.n.
- Pazzani, M., 1996. Searching for dependencies in Bayesian classifiers. *Learning from Data. Lecture Notes in Statistics*, Volume 112, pp. 239-248.
- P, D., Wiratunga, N. & Visweswariah, K., 2012. *Two-part segmentation of text documents*. New York, s.n.
- Porter, M., 1980. *Competitive strategy: Techniques for analysing industries and competitors*. New York: s.n.
- Potrus, M. Y., Ngh, U. K. & Ahmed, B. S., 2014. An evolutionary harmony search algorithm with dominant point detection for recognition-based segmentation of online Arabic text recognition. In: *Ain Shams Engineering Journal*. s.l.:Elsevier, pp. 1129-1139.

- Psychatry, A. A. o. C. & A., 2018. *Social Media and Teens*. [Online]
Available at: https://www.aacap.org/AACAP/Families_and_Youth/Facts_for_Families/FFF-Guide/Social-Media-and-Teens-100.aspx
[Accessed 26 January 2021].
- Qian, J. et al., 2019. *A Benchmark Dataset for Learning to Intervene in Online Hate Speech*. s.l., s.n.
- Qin, Y.-p. & Wang, X.-k., 2009. *Study on Multi-label Text Classification Based on SVM*. Tianjin, s.n.
- Quinlan, J., 1987. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3), pp. 221-234.
- Raganato, A., Bovi, C. D. & Navigli, R., 2016. *Automatic construction and evaluation of a large semantically enriched wikipedia*. s.l., s.n.
- Ranjan, M. et al., 2017. Document Classification using LSTM Neural Network. *Journal of Data Mining*, 2(2), pp. 1-9.
- Razavi, A. H., Inkpen, D., Uritsky, S. & Matwin, S., 2010. *Offensive Language Detection Using Multi-level Classification*. s.l., Springer, pp. 16-27.
- Rocchio, J., 1971. Relevance feedback in information retrieval. In: *The SMART Retrieval System: Experiments in Automatic Document Processing*. NJ, USA: Englewood Cliffs: Prentice-Hall, pp. 313-323.
- Ross, B. et al., 2016. Measuring the Reliability of Hate Speech Annotations : The Case of the European Refugee Crisis. *NLP4CMC III: 3rd Workshop on Natural Language Processing for Computer-Mediated Communication (Bochum)*, Volume 17, pp. 6-9.
- Saha, P., Mathew, B., Goyal, P. & Mukherjee, A., 2018. *Hateminers : Detecting Hate speech against Women*. s.l., s.n.
- Sahgal, D. & Parida, M., 2014. *Object Recognition Using Gabor Wavelet Features with Various Classification Techniques*. Berlin/Heidelberg, s.n.
- Salton, G., 1989. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Boston: Addison-Wesley Longman Publishing Co., Inc..
- Sanjay, G., Nagori, V., Sanjay, G. & Nagori, V., 2018. Comparing Existing Methods for Predicting the Detection of Possibilities of Blood Cancer by Analyzing Health Data. *International Journal for Innovative Research in Science & Technology*, 4(9), pp. 10-14.
- Selvi, S. et al., 2017. *Text categorization using Rocchio algorithm and random forest algorithm*. Chennai, India, s.n.
- Sharifloo, A. A. & Shamsfard, M., 2008. *A bottom up approach to Persian stemming*. s.l., s.n.
- Shushkevich, E. & Cardiff, J., 2018. *Classifying Misogynistic Tweets Using a Blended Model: The AMI Shared Task in IBEREVAL 2018*. s.l., s.n.
- Shushkevich, E. & Cardiff, J., 2018. *Misogyny Detection and Classification in English Tweets: The Experience of the ITT Team*. s.l., s.n.

- Social, S., 2020. *Teen Social Media Statistics 2020 (What Parents Need to Know)*. [Online]
Available at: <https://smartsocial.com/social-media-statistics/>
[Accessed 26 January 2021].
- Soheily-Khah, S., Marteau, P. & Béchet, N., 2018. *Intrusion detection in network systems through hybrid supervised and Unsupervised Machine Learning Process: A Case Study on the ISCX Dataset*. HAL, s.n.
- Sowmya, B. & Srinivasa, K., 2016. *Large scale multi-label text classification of a hierarchical data set using Rocchio algorithm*. Bangalore, India, s.n.
- Suhartono, D., 2014. Lemmatization Technique in Bahasa: Indonesian. *JOURNAL OF SOFTWARE*, 9(5), pp. 1202-1209.
- Thangaraj, M. & Sivakami, M., n.d. Text Classification Techniques: A Literature Review. *Interdisciplinary Journal of Information, Knowledge and Management*.
- Tulkens, S. et al., 2016. *A Dictionary-based Approach to Racism Detection in Dutch Social Media*. A Dictionary-based Approach to Racism Detection in Dutch Social Media, s.n.
- Universitet, U., n.d. *lingfil*. [Online]
Available at: <https://cl.lingfil.uu.se/~nivre/master/NLP-Parsing.pdf>
[Accessed 19 December 2020].
- Vapnik, V. N. & Chervonenkis, A. Y., 1964. A class of algorithms for pattern recognition learning. *Avtomat. i Telemekh.*, 25(6), p. 937–945.
- Vateekul, P. & Kubat, M., 2009. *Fast Induction of Multiple Decision Trees in Text Categorization from Large Scale, Imbalanced, and Multi-label Data*. Miami, s.n.
- Wang, J., Yao, Y. & Liu, Z. J., 2007. *A new text classification method based on HMM-SVM*. Sydney, s.n., pp. 1516-1519.
- Wang, L., Cao, Z., Xia, Y. & De Melo, G., 2016. *Morphological Segmentation*. North America, s.n.
- Wang, Y., Khardon, R. & Protopapas, P., 2016. Nonparametric bayesian estimation of periodic light curves.. *The Astrophysical Journal*, 756(1).
- Wang, Y., Wang, M. & Fujita, H., 2020. Word Sense Disambiguation: A comprehensive knowledge exploitation framework. *Knowledge-Based Systems*, Volume 190.
- Warner, W. & Hirschberg, J., 2012. *Detecting Hate Speech on the World Wide Web*. s.l., s.n.
- Waseem, Z. & Hovy, D., 2016. *Hateful symbols or hateful people? predictive features for hate*. San Diego, California., s.n.
- Wolff, R., 2020. *Applications in Business*. [Online]
Available at: <https://monkeylearn.com/blog/natural-language-processing-applications/>
[Accessed 4 January 2021].
- Wolff, R., 2020. *Text Classification vs Text Extraction: What's the Difference?*. [Online]
Available at: <https://monkeylearn.com/blog/text-classification-vs-text-extraction/>
[Accessed 14 January 2021].

Woolf, B. P., 2009. Chapter 5 Communication Knowledge. In: *Building Intelligent Interactive Tutors*. s.l.:Elsevier, pp. 136-182.

Wu, Y., Zhang, Y., Luo, S.-m. & Wang, X.-j., 2007. *Comprehensive Information Based Semantic Orientation Identification*. Beijing, s.n.

Xia, H., Tao, M. & Wang, Y., 2010. *Sentiment text classification of customers reviews on the Web based on SVM*. Yantai, s.n.

Xiong, D. & Zhang, M., 2014. A Sense-Based Translation Model for Statistical Machine Translation. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, p. 1459–1469.

Zhong, Z. & Ng, H. T., 2012. Word Sense Disambiguation Improves Information Retrieval. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Jeju Island, Korea: Association for Computational Linguistics, p. 273–282.

Zhou, V., 2019. *Building a Better Profanity Detection Library with scikit-learn*. [Online] Available at: <https://victorzhou.com/blog/better-profanity-detection-with-scikit-learn/> [Accessed 20 January 2021].

IX. Appendix A Project Log

Date and time of meeting	12/10/2020
Brief Description of Work done since last meeting:	NONE (First Meeting).
Hours spent on project since last meeting	0
Issues identified during supervision:	NONE (First Meeting).
Agreed tasks for next meeting:	Perform research and produce a detailed sample of the table of content to be used.
Date and time of next meeting	26/10/2020
Student signature	Haiiel-Marie AGBO
Supervisor signature	Usman Butt

Date and time of meeting	26/10/2020
Brief Description of Work done since last meeting:	1. Performed a thorough research on the topic of the dissertation to come up with a potential table of content. 2. Reviewed existing work to pinpoint the gap in the research.
Hours spent on project since last meeting	30
Issues identified during supervision:	Minor changes to the table of content and advise on the way the dissertation should be done i.e.: it should be done following the objectives.
Agreed tasks for next meeting:	Objectively work on the literature review by following the objectives set in the research proposal. Review NLP techniques by being critical.
Date and time of next meeting	16/11/2020
Student signature	Haiiel-Marie AGBO
Supervisor signature	Usman Butt

Date and time of meeting	16/11/2020
Brief Description of Work done since last meeting: 1. Table of content designed . 2. Critical discussion of the NLP techniques 3. Review of existing work and gap in the research identified 4. Clear statement of the aim 5. Review of text classification process and proposed custom process.	
Hours spent on project since last meeting	120
Issues identified during supervision: Lack of criticality while doing the literature review, lack of visual graphs and representation, improvements needs to be done regarding the references, custom text classification needs to meet the requirements of the project.	
Agreed tasks for next meeting: Review existing literature review in order to understand how a critical analysis is performed, include more graphs and references, and proceed to the critical review of classifiers.	
Date and time of next meeting	30/11/2020
Student signature	Haiiel-Marie AGBO
Supervisor signature	Usman Butt

Date and time of meeting	30/11/2020
Brief Description of Work done since last meeting: 1. Discussed text classification stages 2. Inserted more graphs to give a visual representation of the work done 3. Critically analysed NLP techniques and text classification stages by referring to existing work and how they impacted the way each of the stages and techniques function 4. Identify the type of algorithm existing and the way they are used in the classification stage of text classification.	
Hours spent on project since last meeting	96
Issues identified during supervision: beware of plagiarism and appropriate referencing.	
Agreed tasks for next meeting: Discuss identified classifiers and algorithms in a critical manner, do referencing and formatting properly and perform a thorough research on popular datasets used in the hate speech and abusive language classification.	
Date and time of next meeting	14/12/2020
Student signature	Haiiel-Marie AGBO
Supervisor signature	Usman Butt

Date and time of meeting	14/12/2020
Brief Description of Work done since last meeting: 1. Critical discussion of algorithms and classifiers used for text classification 2. Review of the way, algorithms and classifiers have been implemented, improved on and discussion on their limitations 3. Review on how the identified limitations have been covered by various research 4. Documentation of popular hate speech and abusive language datasets in regard to the size, format, authors, and approach.	
Hours spent on project since last meeting	168
Issues identified during supervision: Plagiarism, appropriate referencing, formatting, appropriate representation of the equations related to each classifier, more research needs to be done in the context of limitations. Advise on a summary set of tables to give a graphical overview of the previous discussed methods, techniques, extraction features etc.	
Agreed tasks for next meeting: Review referencing, appropriate formatting of the equations, improvement and sufficient documentation on the classifier's limitations and proposed solutions to the aforementioned limitations. Create a table to compare the feature extraction methods previously discussed and a table to also compare classifiers and algorithms previously discussed.	
Date and time of next meeting	28/12/2020
Student signature	Haiiel-Marie AGBO
Supervisor signature	Usman Butt



Date and time of meeting	28/12/2021
Brief Description of Work done since last meeting: 1. Referencing done properly 2. Comparison of feature extraction methods through the use of a table. The comparison is done by reviewing advantages and limitations 3. Comparison of text classification algorithm and classifiers through the use of a table. The comparison is done by reviewing advantages and limitations 4. Research methodology.	
Hours spent on project since last meeting	96
Issues identified during supervision: NONE	
Agreed tasks for next meeting: Finish the methodology section and begin the implementation stage. Pre-processing of dataset and training of the algorithm.	
Date and time of next meeting	11/01/2020
Student signature	Haiiel-Marie AGBO
Supervisor signature	Usman Butt


Date and time of meeting	11/01/2021
Brief Description of Work done since last meeting:	1. Research Methodology completed 2. Data pre-processing done. 3. Algorithm selected and trained. 4. Implementation complete.
Hours spent on project since last meeting	120
Issues identified during supervision:	Improvement on the layout of the screenshots and description of the code and results. Formatting issues in regard to the recommendation of the handbook.
Agreed tasks for next meeting:	Appropriate documentation of the screenshots of the documentation, complete formatting of the dissertation in regard to the handbook recommendation, finish conclusion and recommendation.
Date and time of next meeting	16/01/2021
Student signature	Haiiel-Marie AGBO
Supervisor signature	Usman Butt

Date and time of meeting	16/01/2021
Brief Description of Work done since last meeting:	Appropriate formatting of the dissertation, editing of the screenshots of the implementation and completion of conclusion and recommendations.
Hours spent on project since last meeting	40
Issues identified during supervision:	NONE
Agreed tasks for next meeting:	NONE
Date and time of next meeting	NONE
Student signature	Haiiel-Marie AGBO
Supervisor signature	Usman Butt

X. Appendix B Ethics Form

My Documents

Amendments						
 Create New Amendment  Refresh						
SUBMISSION ID	CREATED DATE TIME	CREATED BY	STATUS	DESCRIPTION	UPDATED DATE TIME	COORDIN...
No items to display.						

Submission	
Submission Ref	16504
Status	Approved
Submission Coordinator	Hamid Jahankhani hamid.jahankhani@northumbria.ac.uk
Name	<input type="text" value="haiiel-marie.agbo"/>  haiiel-marie.agbo
Email	haiiel-marie.agbo@northumbria.ac.uk
Faculty	<input type="text" value="Engineering and Environment"/>
Department	<input type="text" value="Computer and Information Sciences"/>
Submitting As	<input type="text" value="PGT - Postgraduate Taught student"/>
Externally Approved	<input type="checkbox"/> Note: ONLY tick this box if your project has already received full ethical approval from an external organisation
Module Level Approval	<input type="checkbox"/> Tick this box if staff and this submission refers to an entire module.
Module Code	<input type="text" value="LD7028"/> <input type="button" value="Help"/>
Module Tutor	<input type="text" value="Hamid Jahankhani"/> <input type="button" value="Find"/> <input type="button" value="Help"/> <input type="button" value="Clear"/>
	Titl... Staff at Partner Institute De... Campus Services Em... hamid.jahankhani@northumbria.ac.uk
Research Supervisor	<input type="text" value="Haider Alkhateeb"/> <input type="button" value="Find"/> <input type="button" value="Help"/> <input type="button" value="Clear"/>
	Titl... London Campus Staff De... Vice Chancellors Office

Em... haider.alkhateeb@northumbria.ac.uk

Named Submission
Coordinator (PGT/UGT
only)

hamid.jahankhani@northumbria.ac.uk

Find

Help

Clear

If you are an undergraduate or postgraduate taught student please select a Named Submission Coordinator. If you are not sure who this is please contact your Module tutor or Supervisor as appropriate.

Ethical Risk Level

Low

[Click here to answer the ethical risk questions](#)


Risk Level Conditions:

Your ethical risk is **low**. Your research should only consist of one or more of the following:

- Analysis of secondary data which has been previously published.
- Desk or lab-based research which does not involve collecting data from people (other than pilot data collected solely within the research team).






Your project proposal does not need to be reviewed by your Faculty Research Ethics Committee, however, you need to be ethically aware and ensure that you have not breached plagiarism or copyright regulations and have adequately referenced your material. It is recommended that you refer to Northumbria Research Ethics Policy.

Ethical Risk Diagnostic Questions and Responses

 Refresh

ID	QUESTION	ANSWER
No items to display.		

Co-investigators

 Add  Edit  Delete  Save  Refresh

NAME OF CO-INVESTIGATORS
No items to display.

G1: General Aims and Research Design (Mandatory)

Title

Title of your research project

The role of Machine Learning in the fight against online Anti-Social Behavior.

Outline General Aims and Research Objectives

State your research aims/questions (maximum 500 words). This should provide the theoretical context within which the work is placed, and should include an evidence-based background, justification for the research, clearly stated hypotheses (if appropriate) and creative enquiry.

To develop a method based on machine learning to detect and categorize offensive slang and unacceptable language.

G2: Research Activities (Mandatory)

Please give a detailed description of your research activities

Please provide a description of the study design, methodology (e.g. quantitative, qualitative, practice based), the sampling strategy, methods of data collection (e.g. survey, interview, experiment, observation, participatory), and analysis. Do sensitive topics such as trauma, bereavement, drug use, child abuse, pornography, extremism or radicalisation inform the research? If so have these been fully addressed?

- Run experiments to detect and enhance the detection capabilities of the proposed method.
- Rtools and Python will be utilised to train the algorithms and implements the experiments.
- Perform a literature review of similar work to understand their methodology (secondary data). The analysis of secondary data is the analysis of data previously gathered by another researcher for a certain primary purpose. Using this data provides a credible set of data very useful for researchers who do not have a lot of time and resources.
- Use and improve similar on algorithms and datasets for testing and training purposes.

G3: Research Data Management Plan (Mandatory)

Anonymising Data (mandatory)

Describe the arrangements for anonymising data and if not appropriate explain why this is and how it is covered in the informed consent obtained.

For this research the data will be generated or collected from people (tweets, post, text messages etc...). It is critical to clarify the scope of the research to the people the data is collected from. The data will be used for the sole purpose of this project and deleted after usage. Only data in English is to be collected and studied and the "Information Commissioner's Office" (ICO) is chosen as a convenient "supervisory authority" under the "British data Protection Act" (Legislation.gov.uk, 1998) and the "European General Data Protection Regulation" (Publications.europa.eu, 2016) (ico.org.uk, n.d.).

This research aims to detect abusive languages and with the nature of data to be collected (text messages, tweets, private conversation etc...) it is important to insure the security of the resource and privacy of people. Systems used must be kept in a secure environment (use of firewalls, anti-virus, intrusion detection systems etc...) and data should be encrypted from its original form to its processed form. At the end of the research raw data is to be destroyed and data used out of the working environment should be anonymised.

The project is to propose an algorithm or method based on existing algorithms and datasets in use. It is important to make sure that there is no violation of intellectual property and to adhere to the ethical constraint appropriate to an MSc project.

It is necessary to examine the ethics of setting up an abusive language detection simulation system because it is important to have full consent from people providing the data used by the algorithm to train and improve the accuracy of their detection. Open source simulators will be used because of possible intellectual property rights to be respected. Efforts will be made to generate simulations that are not infringing.

Storage Details (mandatory)

Describe the arrangements for the secure transport and storage of data collected and used during the study. You should explain what kind of storage you intend to use, e.g. cloud-based, portable hard drive, USB stick, and the protocols in place to keep the data secure.

If you have identified the requirement to collect 'Special category data', please specify any additional security arrangements you will use to keep this data secure.

Systems used to keep the data collected must be kept in a secure environment this will be done by the use of firewalls, anti-virus, intrusion detection systems for computer support.

It will also be encrypted or stored on an encrypted virtual hard disk.

If putted on a hard disk or USB device the data will first be encrypted.

Retention and Disposal (mandatory)

☒ I confirm that I will comply with the University's data retention schedule and guidance.

[Research Data Management link](#)

[General Data Protection Regulations including Data Protection link](#)

[Records Retention Schedule link](#)

G4: Research Project Timescale (Mandatory)

Proposed Start Date

28/09/2020



Proposed End Date

12/01/2021



G5: Additional Information

☐ Externally Funded

External Funder



Please give details of your 'other' funder

Agresso Reference

☐ Franchise Programme Organisation

Please give details of your franchise organisation

Type a value

☐ NHS Involvement

Please give details of any NHS involvement

Type a value

☐ Clinical Trial(s)

Please give details of any Clinical Trial(s)

Type a value

☐ Medicinal Products

Please give details of any Medicinal Product(s)

G6: File Attachments

Additional files can be uploaded e.g. consent documentation, participant information sheet, etc.

Please note: It is best practice to combine all documents into one PDF (This avoids the reviewer having to op...

[Go To Attachments](#)

G7: Health and Safety (Mandatory)

☒ I confirm that I have read and understood the University's Health and Safety Policy.

☒ I confirm that I have read and understood the University's requirements for the mandatory completion of risk assessments in advance of any activity involving potential physical risk.

The University Health and Safety Policy can be accessed [here](#)

The University Risk Assessment Code of Practice can be accessed [here](#)

Please confirm either:

☐ There are PHYSICAL risks associated with the research project work and I confirm that a risk assessment has been approved and attached to this ethics submission.

OR

☒ I can confirm that there are no physical risks associated with this project and so no risk assessments are required.

Students requiring assistance with completing their risk assessment should get in touch with their supervisor or module tutor as the first point of contact. If further assistance is needed, the Faculty Technician can provide further guidance.

For more specific risk assessments (e.g. lab work), especially where the project is Medium or High risk, you are required to consult the Faculty Technical Manager; your Supervisor/Module Tutor will be able to put you in touch.

If you have any questions or concerns, please contact the University Health and Safety Team by emailing CRHealthandSafety@northumbria.ac.uk

G8: Insurance (Mandatory)

☐ I have read and understood the University Insurance guidance document (link below):

[Insurance Guidance link](#)

I confirm my work is covered by University Insurance. I confirm an insurance risk level of:

Select an item



If your insurance risk level is HIGH please attach details of exceptional insurance coverage:

[Click here to attach a file](#)

G9: Electronic Signature (Mandatory)

☒ I confirm my supervisor has reviewed the contents of this document

☒ I confirm I have assessed the ethical risk level of my work correctly and answered the above sections as fully and accurately as possible.

Full Name

haiiel-marie.agbo

Date

01 May 2019 14:12:34




PDF Version

Create PDF

No items to display.

Review Comments, Conditions and Outcomes

Log of any Ethical Incidents 

Log New Incident


INCIDENT...	CREATED DATE TIME	CREATOR NAME	COMPLAINANT DETAILS
No items to display.			
Title and Objectives (see G1)			
<div> <div>+ Add</div> <div>Save</div> </div>			
Reviewer A: <input type="text"/>		Reviewer B: <input type="text"/>	
e.g. Are the research question and/or study aims clear?			
DATE	ROLE	COMMENT	
No items to display.			
Proposed Methodology and Analysis (see G2)			
<div> <div>+ Add</div> <div>Save</div> </div>			
Reviewer A: <input type="text"/>		Reviewer B: <input type="text"/>	
e.g. Is the design appropriate to the research question?			
Are the methods of data analysis appropriate to the research question?			
DATE	ROLE	COMMENT	
No items to display.			
Sample and Recruitment (see M1)			
<div> <div>+ Add</div> <div>Save</div> </div>			
Reviewer A: <input type="text"/>		Reviewer B: <input type="text"/>	
e.g. Is the sampling approach appropriate to the design?			
Is the sample sufficient and achievable?			
Is the process of recruitment clearly explained?			
Are participants receiving payments for taking part, and if so is the payment appropriate?			
If the DBS is ticked, has the appropriate information been included?			
DATE	ROLE	COMMENT	
No items to display.			
Consent (see M1)			
<div> <div>+ Add</div> <div>Save</div> </div>			
Reviewer A: <input type="text"/>		Reviewer B: <input type="text"/>	
e.g. Is the approach to consent seeking clear?			
Is consent from parents/ carers/ guardians required?			
Are all necessary recruitment and informed consent documentation included (e.g. letters of permission, letters of invitation)			
Is the information sheet adequate to ensure informed consent?			
Are the consent form(s) appropriate?			
DATE	ROLE	COMMENT	
No items to display.			
Researcher and Participant Safety (see M1)			

 Add
  Save

Reviewer A:
 Reviewer B:

e.g. Is there any risk of physical harm for the researcher(s) or the participants and if so what attempts have been made to alleviate or minimise them?
Have Risk Assessments been referred to where appropriate?

DATE	ROLE	COMMENT
No items to display.		


Research Activities (see G2-G8, M1-M5, H1-H5)




 Add
  Save

Reviewer A:
 Reviewer B:

e.g. Are the research tasks described clearly?
Do sensitive topics such as trauma, bereavement, drug use, child abuse, pornography or extremism/ radicalism inform the research? If so have these been fully addressed? (and we can use this to amend the information on risk levels on the form)Is there any risk that the tasks may cause psychological harm and if so what attempts have been made to alleviate or minimise them?

DATE	ROLE	COMMENT
No items to display.		


Data Management Plan (see G3)




 Add
  Save

Reviewer A:
 Reviewer B:

e.g. Have sufficient steps been taken to ensure participant anonymity/ confidentiality of data?
Are the arrangements for data storage and disposal clearly outlined?
Are these arrangements in line with University and/or the funding body requirements?

DATE	ROLE	COMMENT
No items to display.		


File Attachments (see G6)


 Add
  Save

Reviewer A:
 Reviewer B:

Please note: where file attachments have not been added because they are not required, please select Approve.

COMMENT BY	DATE	ROLE	COMMENT
No items to display.			

General Comments (see Help)


 Add
  Save
 

DATE	ROLE	COMMENT
No items to display.		