Machine Learning for Fascist Text Classification in Social Networks

Dissertation in MSc Computer Science

Author: Siôn William Davies

ID: 1349541

Supervised by Dr. Tom Chothia

University of Birmingham

September 2020

Abstract

In recent years much research has been devoted to the automated prevention and monitoring of hate speech in social networking platforms. This has predominantly focused upon the capabilities of machine learning systems to fulfil this objective. The need for such systems has largely been associated to the growing influence that social networking services hold over modern society. Concurrently with this trend, we are also living in a time that has seen a marked shift in the rise of extremist ideologies around much of the world. Perhaps the most pronounced manifestation of which is the extremist right-wing faction known as the 'Alt-right'. Some political commentators have also attributed the rise of such factions to the increased surge in online connectivity, arguing that the internet has provided a concealed platform for many extremists to circulate their ideologies and target victims. Consequently, terms such as 'digital-fascism' and 'cyber-fascism' have been coined to describe this development. This research looks to propose a solution to combat this unfavourable trend, by exploring the abilities of machine learning techniques to classify what we define to be specifically fascist speech. In order to achieve this, we create four distinctly diverse datasets containing examples of fascist speech through both manual annotation and data augmentation methods. In our first area of research, we evaluate a range of machine learning algorithms (Logistic Regression, Random Forest, Support-vector Machine) and feature extraction techniques (word embeddings, Paragraph vector, TF-IDF) in their ability to classify binary fascist vs. non-fascist speech. From this investigation, we select our most successful models to progress forward to a multiclass classification experiment, in which we incorporate hate speech as an additional class. In this second investigation, it is our intention to establish as to whether or not we can provide evidence to demonstrate a distinction between the two styles of languages concerning fascist and hate speech. The results produced in this research demonstrates the superiority of the Support-vector Machine, which we implement in the form of a Linear-SVC (Support-vector Classifier), above the alternative classification algorithms. We further show the effect that data quantity has upon the performance of word embedding and Paragraph vector features. Our best performing multiclass classifier was implemented in the form of a Linear-SVC with TF-IDF character n-gram features, achieving F1 scores of 0.93% and 0.88% for the classification of fascist and hate speech respectively.

Keywords:

Machine Classification; Supervised Learning; Natural Language Processing; Hate speech; Support Vector Machine; Logistic Regression; Random Forest; TF-IDF; Word embeddings; Paragraph vector; Synonym Replacement; Sentence reordering; SMOTE

Acknowledgements

I would like to thank Dr. Tom Chothia for proposing this project and for his excellent supervision and guidance throughout its duration. I also wish to thank my parents for their continual support in all my endeavours.

Table of Contents

List of Figures

List of Tables

CHAPTER 1: Introduction

1.1 Motivation

According to a 2020 study published by Statista, the global penetration rate for social networking sites currently stands at 49%, with worldwide users spending an average of 144 minutes a day connecting with one another on social networks during 2019 (Statista, 2020). This was an increase of 62.5% since 2012, roughly equivalent to an additional hour of usage per day. Consequently, it is unsurprising that this surge in online connection has coincided with an increase in the volume of online hate crimes and prejudiced experienced by users of such platforms.

As a result, the need to moderate inappropriate conduct on social networks has become a hot topic in recent years. Mainly, hate speech systems have been increasingly deployed by companies seeking to control these issues in conjunction with the manual moderation process. Nevertheless, these systems are far from the finished product and have come under scrutiny for their shortcomings. Facebook CEO Mark Zuckerberg stated in 2018 that it would still likely take between five to ten years before AI (Artificial Intelligence) systems could master all the linguistic subtilties comprising hate speech (The Washington Post, 2018).

In concurrence with the increase of online hate crimes, we are observing a rising shift in the presence of extremist ideologies in Europe. Traverso (2019) draws to attention that as of 2018, eight notable countries are governed by far-right parties who exhibit ‘nationalist’ and ‘xenophobic’ tendencies (Austria, Belgium, Denmark, Finland, Italy, Poland, Hungary and Slovakia). Moreover, many other regions outside of Europe have also felt these effects and perhaps none more prevalently than the United States. The ascension of Donald Trump to the White House arose from a campaign trenched in right-wing populist themes, which focused predominantly on exploiting citizens fears of immigration. According to Foster (2017), Trump has instilled a neo-fascist administration that has parallels to the traditional fascism observed in Italy and Germany in the early Twentieth Century.

At the time Foster was writing about this, the world witnessed the re-emergence of far-right movements in the US. One of the most publicized cases of this was witnessed globally in the ‘Unite the Right Rally’, conducted by self-proclaimed neo-Nazi and white supremacists in Charlottesville, Virginia in 2017. Such behaviour resulted in political commentators declaring that a new wave of neo-fascism was emerging on a world-wide scale.

The catalyst driving this re-emergence of fascist philosophies is most likely the combination of several contributing factors. Nonetheless, one could argue that a medium that is greatly propelling the circulation of fascist ideals is online social networking communities. Such an assessment is not a new concept however, as twenty years ago the term ‘Cyber-fascism’ was coined by Griffin (2000) to describe just this. Recent digital advancements and internet availability has been capitalized on by far-right factions in particular, who have sought to use it as a platform to connect, spread and advertise fascist ideals to adherents and the susceptible alike (Daniels, 2009). Fielitz and Marcks (2019) argue that social networking has provided a ‘beneficial terrain’ for the far-right who are using ‘digital fascism’ as a means to endanger ‘open societies’. They highlight the conundrum of desiring a society

which promotes freedom of speech, and yet, by doing so we enable the proliferation of concepts such as digital fascism to threaten our democratic principles.

The most palpable manifestation of this modern wave of post fascism has presented itself in a movement referred to as the 'Alt-Right'. Common characteristics of the Alt-Right include the opposition to gender equality, the promotion of a patriarchal society and at its core the belief in the superiority of the white race (Hawley, 2017). One feature of the Alt-right movement is its predominantly online presence. *The Daily Stormer* and the *Renegade Tribune* are just two examples of neo-fascist media networks who align themselves with the Alt-Right, with the former advocating for the genocide of Jewish community (The Huffington Post, 2018).

Not all online fascist commentary is so overtly published on unconcealed networks. Social network conglomerates such as Facebook and Twitter have experienced serious problems trying to extinguish fascist content from their platforms. Facebook was recently embroiled in a legal dispute for banning the account of Italian neo-fascist political party *CasaPound* from their site, which had roughly 240,000 followers (The Guardian, 2019). The backlash this decision was greeted with demonstrates the ambiguousness regarding what is and isn't acceptable language to be circulated on social networks. What is widely accepted, however, is the need to curtail the spread of fascist content to protect those it seeks to prey on. Hopefully we can agree that the use of fascist ideology to subjugate others has no place in a modern society that promotes equality for all as an essential human right.

1.2 Aims and Research Questions

Machine Learning systems have been modelled to have the ability to automatically detect hate speech in social networks. This research concerns attaining a similar objective. Principally, we want to discover if we can use alike machine learning techniques to correctly classify what we will consider to be fascist speech. To achieve this, we will be combining the fields of machine learning and natural language processing (NLP). As a sidenote, we use the terms 'fascist speech' and 'fascist text' interchangeably in this study.

To be able to train a machine learning model to recognize fascist speech, we will foremost have to provide a working definition that we consider accurately reflects contemporary fascist speech. From this definition, we will look to acquire a sufficient number of data samples containing fascist speech to be able to train a robust machine learning model. In addition to this we aim to identify, implement and evaluate a range of feature extraction techniques and machine learning algorithms in their ability to classify fascist speech. In the process, we seek to answer the following questions:

- *What affect does the choice of feature extraction technique have on a machine learning model's ability to classify fascist speech?*

- *What affect does the choice of machine learning algorithm have on a machine learning model's ability to classify fascist speech?*

- *What affect does the size and quality of the dataset have on a machine learning model's ability to classify fascist speech?*

In order to ascertain answers to the above questions, we carry out two classification investigations. These investigations attempt to answer the questions:

- *Can we build a binary machine learning model that can, to some degree of accuracy, classify fascist vs. non-fascist speech?*

- *Can we build a multiclass machine learning model that can, to some degree of accuracy, classify fascist vs. hate speech?*

Through answering the latter of the above questions, we further pursue an answer to the question:

- *Can we use machine learning techniques to prove that there is (or isn't) a clear distinction between fascist and hate speech?*

1.3 Structure of the Report

This research is extended into eight subsequent chapters, comprising nine in total. This chapter has provided the motivation for this research, in addition to the questions it seeks to answer. We divide the remainder of this study into the following chapters…

Chapter 2: Provides a review of the related literature concerning hate speech classification. We discuss some of the most recent findings in this field to provide the context for our own work.

Chapter 3: Introduces some important background concepts related to this area of research.

Chapter 4: Seeks to formulate a working definition of fascist speech. To achieve this, the chapter provides a historical context of fascism and further consults some leading political and historical commentators' opinions on the subject matter.

Chapter 5: Details the dataset selection process.

Chapter 6: Discusses the methodology that was executed in this study, including data pre-processing; feature extraction; model selection and hyperparameter optimization.

Chapter 7: Explores the binary fascist vs. non-fascist classification experiments. We discuss the results and evaluate our approaches.

Chapter 8: Takes the best performing models identified from chapter 7 into a multiclass classification investigation in which hate speech is integrated as an additional class. In addition to analysing the results, this chapter also attempts to infer whether we can demonstrate a distinction between the two styles of languages.

Chapter 9: Concludes the research. We discuss our contributions and achievements whilst also addressing the study's limitations.

CHAPTER 2: Literature Review

Introduction

In this chapter, we discuss the literature of the recent work concerning hate speech classification. Hate speech classification literature was chosen to review due to it being the most similar field to this research's task. Section 2.1 covers the subject of data, including data collection, annotation, biases and augmentation. Section 2.2 lists some of the most popular methods to achieve feature representation. Finally, section 2.3 provides a tabular overview of the most frequently occurring machine learning algorithms that appear in the literature and discusses a select number of their implementations.

2.1 Data Selection

When using machine learning solutions to overcome a task, the provision of more data samples to train a model will more often than not lead to a superior classifier (Domingos, 2012). As such, the more examples we can provide the better our chances of training a successful model, and, the decreased chances we have of overfitting (see 3.4).

The majority of data sourced for hate speech research has been derived directly from social media posts. Sources of which can be viewed in table 1. Most studies gathered their data by searching for key words, hashtags or scraped data from periods where notable events occurred (Burnap and Williams, 2015).

Data Source	Study
Facebook	(Del Vigna et al., 2017)
Twitter	(Go et al., 2009), (Burnap and Williams, 2015), (Bravo-Marquez et al., 2016), (Waseem and Hovy, 2016), (Badjatiya et al., 2017), (Davidson et al., 2017), Gambâck and Sikdar, 2017), (Malmasi and Zampieri, 2017)
Stormfront	(Gitari et al., 2015)
Yahoo!	(Warner and Hirschberg, 2012), (Djuric et al., 2015), (Nobata et al., 2016)

Table 1: Data sources and their respective studies.

2.1.1 Data Annotation

Various data annotation methods have been applied. For some studies the appropriate pre-labelled datasets are readily available for use (Badjatiya et al., 2017), (Malmasi and Zampieri, 2017). When this is not possible, many researchers have taken to the task of labelling their own data (Gitari et al., 2015). Alternatively, some researchers have hired external sources to annotate data on their behalf, providing the annotators with a description of the categories to label (Warner and Hirschberg, 2012), (Waseem and Hovy, 2016). Whereas other studies have opted to use Crowdsourcing (Burnap and Williams, 2015), (Gambâck and Sikdar, 2017).

Previous research has highlighted disadvantages with the human process of manually annotating data. The most obvious one is the time consumed by trawling through thousands of samples and the cost that this brings. Moreover, the subjectivity of the annotator is also in question. This was

demonstrated by Del Vigna et al. (2017) who computed a Fleiss' kappa inter-rate agreement metric among their annotators. They found that among five student annotators classifying Facebook comments into categories of 'no-hate', 'weak hate' and 'strong hate', their *k* metric indicating the level of agreement only totalled 26% over the three classes.

Nonetheless, Durate et al. (2018) state that manual supervised annotation performed in accordance with a clear and precise definition of the speech to be classified will produce the most accurate results. This they argue, is more preferable than using automated technologies which cannot be relied upon to capture the 'nuanced analysis' of the speakers' true intentions, especially concerning social media posts.

2.1.2 Data Biases

When selecting data for classifier training purposes we must be vigilant, as biases that are present in the corpus (collection of text data) can be amplified by the model it produces. For example, Dixon et al. (2018) established a relationship between comment length and toxicity classification, with comments of a shorter length more likely to be toxic. This, they note, would create a classifier that would almost always associate a shorter comment that possessed an identity term as being inherently toxic. They further found that identity terms that occurred heavily in the same category within the training data led to a classifier incorrectly over-generalizing on the same terms when evaluated on real-world data.

Minority classes and imbalanced training data can also result in a model that is biased in favour of the majority class. To address this concern, Indurthi et al. (2019) proposed the use of SMOTE (Synthetic Minority Over-sampling Technique) in their study. This technique generates artificial data instances from original samples until any class imbalances have been offset. A model they produced in association with this technique, that sought to classify hate speech against immigrants and women on Twitter achieved a 0.65% accuracy score.

2.1.3 Data Augmentation

Data augmentation is a technique that is used across a wide range of machine learning tasks. The process of data augmentation involves modifying existing data to create new augmented versions from it. This can be extremely useful for classification problems in which we find ourselves with certain classes that are adversely imbalanced, and if performed well, this allows us to increase the diversity of data we can provide to train a model.

Data augmentation techniques were noted as having a positive influence in certain hate speech studies. Gröndahl et al. (2018) augmented their training set by means of stochastic transformation (swapping words based on a random probability distribution), allowing them to double their quantity of data. This also increased the diversity of words their model had to train on, which they stated diminished the impact that 'benign' (non-hateful) words had on their classifier.

Zhang et al. (2015) preserved the syntactic and semantic significance of words in their study by replacing words and phrases with their synonyms. Using a WordNet thesaurus, they used a geometric probability distribution that determined the chances of a synonym being chosen to replace a word. The probability of being chosen decreased the further away the synonym was determined to be from the original meaning of the word. They also noted that data augmentation in NLP tasks was extremely

useful when training a deep learning neural network, as they found that the combination of these techniques can help to mitigate against generalization errors.

2.2 Feature Representation

After data has been collected and pre-processed, it must be converted into a machine-readable format. The method used to accomplish this is called 'feature extraction', and it determines how we 'represent' a text input through the means of mapping raw text data to numerical feature vectors. This section discusses some of the commonly applied feature representation techniques reviewed in the literature. A summary of the feature representation techniques identified with their associated papers can be viewed in table 2.

Feature	**Study**
Bag of Words	(Burnap and Williams, 2015), (Djuric et al., 2015), (Badjatiya et al., 2017)
Brown Clustering	(Warner and Hirschberg, 2012), (Malmasi and Zampieri, 2017)
Character n-grams	(Nobata et al., 2016), (Waseem and Hovy, 2016), (Badjatiya et al., 2017), (Del Vigna et al., 2017), (Gambäck and Sikdar, 2017), (Malmasi and Zampieri, 2018)
Dependency Relationships	(Burnap and Williams, 2015), (Del Vigna et al., 2017)
Lexicons	(Burnap and Williams, 2015), (Gitari et al., 2015), (Bravo-Marquez et al., 2016), (Davidson et al., 2017), (Del Vigna et al., 2017)
Paragraph embeddings	(Warner and Hirschberg, 2012), (Djuric et al., 2015)
POS	(Warner and Hirschberg, 2012), (Bravo-Marquez et al., 2016), (Davidson et al., 2017), (Del Vigna et al., 2017),
Sentiment Analysis	(Go et al., 2009), (Gitari et al., 2015), (Bravo-Marquez et al., 2016), (Del Vigna et al., 2017)
TF-IDF	(Badjatiya et al., 2017), (Davidson et al., 2017)
Word embeddings	(Badjatiya et al., 2017), (Del Vigna et al., 2017), (Gambäck and Sikdar, 2017), (Indurthi et al., 2019)
Word n-grams	(Burnap and Williams, 2015), (Waseem and Hovy, 2016), (Davidson et al., 2017), (Del Vigna et al., 2017), (Malmasi and Zampieri, 2017)

Table 2: Feature representation methods and their respective studies.

2.2.1 Surface-level Features

The most common 'surface-level' features for text classification systems include Bag-of-Words (BOW) and n-gram representations. BoW counts the instances of tokens (see 6.1.1) in a given document, discarding their order but maintaining multiplicity, where the occurrence of each word is used to train a classifier. The downside to such an approach is that the contextual meaning of sentences is not taken into consideration. N-gram features propose a solution to this where *n* stands for the number of words within a sequence, and such features can be used to take the surrounding context of a target word into consideration. Commonly observed word n-gram implementations include unigrams, bi-grams and tri-grams.

One disadvantage of this is when encountering misspelled words. Character n-grams address this problem as they can capture resemblances between tokens based on recognized spelling similarities. Kanaris et al. (2006) demonstrated that character n-grams can be more accurate then word-token features in the case of spam classification, even when increasing the dimensionality of the task. Although word and character n-gram features are noted as having high success rates by themselves, when combined with higher-level features they have been found to have enhanced results (Schmidt and Wiegand, 2017).

2.2.2 Word and Paragraph embeddings

Embedding representations help to overcome a data sparsity problem faced by BoW and n-gram features. For example, text classification in social networks is often carried out on smaller sized documents which may only be comprised of a single sentence or a couple of words in some instances. This can cause issues when adopting n-gram features, as the sparsity of data may result in the document not being of an adequate word count for the n-gram representation. To counter issues such as this we can utilize embedding techniques, such as word, sentence and paragraph-level embeddings.

Word embeddings can be used to map words to numeric vector inputs and are commonly generated by neural networks. They are formed in the manner to provide words that share semantic and contextual similarities with an alike representation to one another within the vector space (Mikolov, 2013). They are most notably used in classification tasks that place a heavy focus on the semantic retrieval of text. Badjatiya et al. (2017) used the GloVe word embedding algorithm in their research concerning deep learning for hate speech, whose features earned a 0.886% F1-score in one particular implementation.

As text classification focuses on classifying a set of words in a document, as opposed to individual words, when applied in this context the individual documents themselves can also be represented in the vector space. Djuric et al. (2015) explored this using the paragraph2vec algorithm (also known as Paragraph vector) to form comment-level (document) embeddings. Their binary neural language classifier with comment embedding features achieved an 0.80% accuracy score in the detection of hate speech, out-performing their alternative BoW models. Similarly, Indurthi et al. (2019), opted for sentence-level embeddings in their study. Their best performing model used pretrained Universal Encoder sentence embeddings for feature representation combined with a Support-vector machine using an RBF kernel, which attained an overall F1 score of 0.65%.

2.2.3 Brown Clustering

Brown clustering was the most frequently observed clustering algorithm in the literature (Brown et al., 1992). Brown clusters represent features as a hierarchical binary tree, where each path in the tree conveys a semantic similarity between each node, which in turn are distinctive clusters of words. Words are grouped into clusters primarily based upon their context, with each set of classes implementing a binary merging criterion (Malmasi and Zampieri, 2017). Notably, brown clustering algorithms have achieved good results when used in supervised learning tasks (Warner and Hirschberg, 2012).

2.2.4 Lexicon Approaches

Lexicon approaches make the assumption that the presence of certain target words within a document will indicate its underlying sentiment. Within the context of hate speech studies for instance, it is assumed that the presence of terms denoting specific curse words or offensive slurs in a document

greatly increase the probability of it being classified as hate. As such, an external library referred to as a lexicon can be utilized to assign weights to each word to evaluate the chances that it may belong to a specific category.

Esuli and Sebastiani (2006) generated their own publicly available lexical resource, 'SentiWordNet', using a semi-supervised approach combined with lexical database WordNet semantic relationships. Through this they were able to assign documents numerical scores to determine whether they were positive, negative or objective in relation to their sentiment scores. Gitari et al. (2015) created a lexicon generated via semantic feature expressions and subjectivity detection that they used to build their classifier. They first singled out sentences that they deemed as containing sentiments of an objective nature, as they explain that hateful language is interlaced with very subjective expressions. Subsequently, they used a rule-based method to create a lexicon of hate words using the subjective phrases they had identified, which their classifier then trained from to form its predictions.

2.3.5 Linguistic Features

Part of Speech Tagging (POS), also referred to as grammatical tagging, is used to tag each token in a document to a part of speech, such as a noun, adjective or verb. This tagging is assigned based on the words definition and the context it is composed in. The usefulness of POS to enhance tokens linguistically for text classification tasks has been shown to be limited however, especially in comparison to textual features (Moschitti and Basili, 2004).

Specialized linguistic features can also be used to counter noise found in data. We can implement them in a general sense, or, we can specialize them in accordance with the given task. Nobata et al. (2016) developed their linguistic features to conform to the latter of these options. They formed specialized linguistic features to identify 'inflammatory' and 'politeness' words within documents. Examples of these features included; length of comment in tokens; average length of words; number of URLs and the number of inflammatory or politeness words in a comment.

Typed dependency relationships can also be useful for text classification. Typed dependency relationships deploy the logic that non-consecutive words used in the same document may hint at its inner context (Schmidt and Wiegand, 2017). For example, if we created a dependency of {'Jew', 'Shylock'} then a document containing both of these words should flag as fascist speech (in our research). This is due to the presence of the target group and an offensive religious slur against this target, denoting the likely implication of derogatory anti-Semitic language.

2.3 Classification Algorithms

In this section, we present a selection of the most widely applied machine learning algorithms in the reviewed literature and discuss some of their implementations. Table 3 displays an overview of the frequency of the classification algorithms in their relevant papers. Table 4 conversely, conveys some notable scores for a select number of these algorithms. It was decided not to discuss the scientific theory behind each of the algorithms in this section. Instead, we later do so for the algorithms that were selected for use in this study (see 6.3).

Classification Algorithm	Study
Convolutional Neural Network	(Badjatiya et al., 2017), (Gambâck and Sikdar, 2017)
Decision Trees	(Burnap and Williams, 2015), (Davidson et al., 2017)
Naïve Bayes	(Go et al., 2009), (Kwok and Wang, 2013), (Davidson et al., 2017)
Logistic Regression	(Djuric et al., 2015), (Waseem and Hovy, 2016), (Badjatiya et al., 2017), (Davidson et al., 2017)
Random Forest	(Burnap and Williams, 2015), (Davidson et al., 2017)
Recurrent Neural Network	(Badjatiya et al., 2017), (Del Vigna et al., 2017)
Support-vector Machine	(Moschitti and Basili, 2004), (Go et al., 2009), (Warner and Hirschberg, 2012), (Burnap and Williams, 2015), (Davidson et al., 2017), (Del Vigna et al., 2017), (Malmasi and Zampieri, 2017), (Indurthi et al., 2019)

Table 3: Classification algorithms and their respective studies.

Classification Algorithm	Score	Study
Convolutional Neural Network	0.78 (F1)	(Gambäck and Sikdar, 2017)
Naïve Bayes	0.83 (Acc.)	(Go et al., 2009)
Logistic Regression	0.90 (F1)	(Davidson et al., 2017)
Random Forest	0.77 (F1)	(Burnap and Williams, 2015)
Recurrent Neural Network	0.93 (F1)	(Badjatiya et al., 2017)
Support-vector Machine	0.68 (F1)	(Indurthi et al., 2019)

Table 4: Notable algorithm performances.

2.3.1 Model Implementations

Go et al. (2009) tested a multinomial Naïve Bayes model when researching sentiment classification for tweets. They trained their model with emoticon data using distant learning techniques and managed to obtain 0.83% accuracy score when combined with unigram and bi-gram features.

Warner and Hirschberg (2012) produced an anti-Semitic speech classifier, electing for the SVM algorithm implemented with a linear kernel function. They produced 4379 features using extraction techniques such as POS and brown clusters in a template-based methodology, achieving an F1 score of 0.63%.

Davidson et al. (2017) tested a variety of machine learning algorithms in their study on automated hate speech detection. They used a grid-search (5-fold cross validation) to configure optimal parameter settings and found Logistic Regression and a Linear SVM outperformed other algorithms. For their final model they opted for Logistic Regression configured with the L2 regularization technique, using a 'one-versus-rest' framework to train individual classifiers for each specific class. Their model achieved a terrific F1 score of 0.90%.

Burnap and Williams (2015) used a Random Forest implementation of a Decision Tree (RFDT) in their study, specifying that this was because rule-based approaches to the classification of 'antagonistic' content such as hateful language had proved successful in past research. Their RFDT

model with n-gram reduced type dependency features secured a 0.77% F1 score, which tied results with their SVM model of the same implementation.

Artificial neural networks (ANNs) have also been used in model implementations concerning hate speech classification. Principally, this was noted in the form of Convolutional (CNN) and Recurrent (RNN) neural networks. Gambäck and Sikdar (2017) created a CNN classifier combined with word embedding features created from the Word2vec algorithm. This model attained a 0.78% F1 score, which outperformed their Logistic Regression with character n-grams model in the same hate classification task. Contrarily, Del Vigna et al. (2017) implemented their classifier as an RNN in the form of a Long short-term Memory (LSTM) network, this was due to its capability to capture 'long-term' word dependencies in a given document. In their binary hate vs. no-hate investigation, their LSTM classifier achieved an accuracy score of 0.80%.

CHAPTER 3: Background

Introduction

This chapter introduces several important concepts that are related to this study. The initial subchapters (3.1 - 3.4) introduce background concepts with the intention of enhancing the readers knowledge of the research area. The chapter ends (3.5) by describing the performance metrics that are later used to evaluate our models in chapters 7 and 8.

3.1 Machine Learning

Machine learning is the… "*field of study that gives computers the ability to learn without being explicitly programmed*" (Samuel, 1959). More specifically, it is the study of computer algorithms which can be applied to train mathematical models that have the ability to interpret, learn from and utilize data. As such, it is the process whereby a computer program learns from data. The main types of machine learning approaches are:

I. *Supervised Learning*: Is the process in which a model trains and validates on input-output pairs consisting of labels and samples that have been pre-annotated. Through this means, the model can learn how to map certain inputs to a particular output based on the labelled data it has learned from. Examples of common supervised learning algorithms include Logistic Regression and Naïve Bayes. The investigations conducted in this research principally concern the area of supervised learning.

II. *Unsupervised Learning*: Occurs when the data that is provided to a model is not labelled into categories. In order to learn information from the data, an unsupervised learning model must analyse the structure of the data, extracting the most beneficial features from it. Clustering algorithms such as k-means and hierarchal cluster analysis are popular unsupervised learning approaches.

III. *Semi-supervised Learning*: Involves combining labelled and unlabelled data to train a model. This is performed by training a model on the sampled portion of data, and then having the model make its predictions on the unlabelled samples whilst simultaneously labelling them in the process.

3.2 Machine Learning Process

Figure 1 illustrates the underlying process that is required to build a machine learning model.

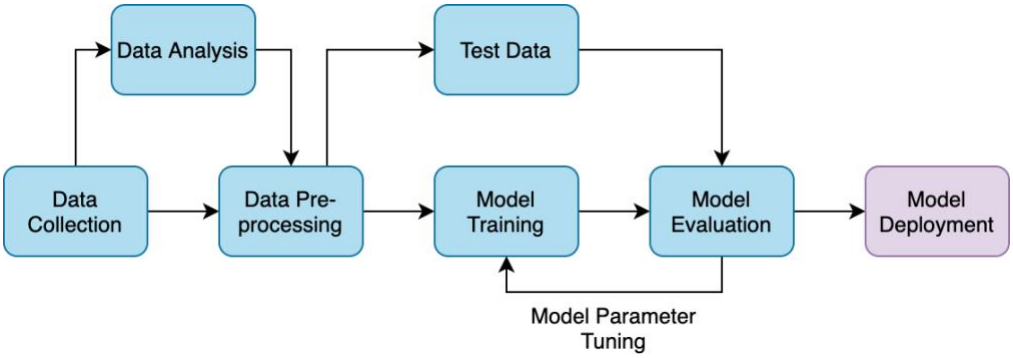

Figure 1: Machine Learning Process

3.3 Natural Language Processing

Natural language processing (NLP) combines the fields of computer science, artificial intelligence and linguistics. Mainly, it concerns how we are able to process and analyse data relating to human languages (i.e. textual data) via computer programmes. Text data comes in a variety of different languages and is naturally unstructured in nature. NLP encompasses a range of different methods that help us to establish a structure for raw text data.

3.4 Overfitting

When used in association with machine learning, overfitting refers to the concept in which a model has been fitted too closely to its training data and it is often caused by a model that is overly complex in nature. An overfitted model will likely result in a poor quality of prediction due to its failure to generalize to new data instances. Popular methods to combat overfitting include cross-validation (see 6.4.1) and the addition of regularization terms in the evaluation function (Domingos, 2012).

3.5 Performance metrics

In order to measure the quality of our trained models, several performance metrics were used. Such metrics are intended to convey how a model might perform in real world applications. In this subchapter we specify which metrics were selected and describe their applications.

3.5.1 Confusion Matrix

Confusion Matrices provide an excellent visual representation of a model's performance. Supervised classification denotes that each class has a true condition and a predicted condition. From this, we can infer that an input variable has four possible outcomes. Those data instances predicted to their correct class are counted as true positives (TP) and true negatives (TN) by the matrix. Conversely incorrect assignments are counted as false positives (FP) and a false negatives (FN). We illustrate the layout of a binary classification confusion matrix in table 5.

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Table 5: Confusion Matrix.

3.5.2 Accuracy

Accuracy is a widely used evaluation metric. When applied to classification, it describes the total number of correct predictions made divided by the total number of predictions. Accuracy can provide a meaningful insight into a model's performance when we have balanced classes in our dataset. Contrarily, if dealing with vastly imbalanced data classes its usefulness is diminished. We calculate accuracy as:

$$A = \frac{(TP + TN)}{(TP + FP + FN + TP)} \tag{3.1}$$

3.5.3 Precision, Recall and F1-score

Precision

Precision is a model's ability to identify *only* the correct data instances belonging to a class. The formula for precision is calculated as:

$$P = \frac{(TP)}{(TP + FP)} \tag{3.2}$$

Recall

Recall focuses on a model's capacity to identify *all* of the correct data instances belonging to a class. Therefore, if we had two target variables 0 and 1 and our model predicts only 1s, it will achieve a 100% recall score. We calculate recall as:

$$R = \frac{TP}{(TP + FN)} \tag{3.3}$$

F1-score

Precision and Recall often cause a trade-off between one another. Therefore, we infuse these two metrics together by taking their harmonic mean to create a third (Portilla, 2019). This is referred to as the F1-score and is usually considered as the most meaningful evaluation metric concerning text classification (dependent upon the data distribution). Its formula is:

$$F1 = 2 * \frac{\mathrm{P} * \mathrm{R}}{\mathrm{P} + \mathrm{R}} \tag{3.4}$$

By using the harmonic mean of precision and recall, the F1-score seeks to punish extreme values. As such, if our model has a great precision score but terrible recall, the F1-score will reflect the poor recall score.

CHAPTER 4: Defining Fascism

Introduction

To be able to acquire suitable data samples containing fascist text, we must first define what it is that we will be attempting to classify. This chapter aims to provide a working definition for what we determine to constitute fascist speech. To achieve this, the chapter will begin by providing some historical context on the subject matter (4.1). Next, we will look to a number of leading political and historical commentators in order to form our definition of fascist speech (4.2). Finally, we look to some prominent definitions of hate speech to ascertain its contrast to fascist speech (4.3).

4.1 Historical context and Neo-fascism

'Fascio' is the Italian word from which the term fascism is derived. Originating in Northern Italy in 1919, Benito Mussolini is credited as the founder of the fascist movement as a political force, which resulted as part of a response against the Enlightenment movement. It was an ideology deeply entrenched with anti-democratic and anti-liberal values. The end of the First-World War created the perfect environment for its ideals to grow, with those nations on the losing side being heavily punished and consequently finding themselves feeling aggrieved and impoverished. Having both served their countries in the War, Benito Mussolini and Adolf Hitler formulated their own political movements that were steeped in ideologies of war, violence, right-wing nationalism and imperialism (Finchelstein, 2017).

Mussolini's *fascisti* party first came to prominence in Italy in the 1920s to the vast approval of the Italian nation. Fascism spread quickly throughout Europe and there wasn't a single European country that didn't spawn a copycat fascist party of some form thereafter (Thompson, 2011). Shortly afterwards in 1933, Hitler and the *Nazi* party seized power in Germany and forged a totalitarian state. Mussolini and Hitler were dictators whose reigns had many similarities yet there were some distinctions in their implementations of the ideology. For example, Nazi Germany made race and anti-Semitism a top priority on their agenda, however, the core principles of their states remained similar. The end of the Second-World War and the victory of the Allied powers resulted in the suicide of Hitler and the execution of Mussolini by Italian Partisans. This was the end to what we refer to as 'classical fascism' (Pauley, 2014).

'Neo-fascism' is the term that is used to refer to the post War ideology that ensued from classical fascism. As such, it encompasses the shadow of the former fascist philosophy that has endured into modern society. Since the 1970s fascism has experienced a revival. It was around this time that notable fascist parties such as the *National Front* (NF) in Britain, the *Front National* in France (FN) and the *Italian Social Movement* (MSI) in Italy all experienced some degree of success. Within Europe fascist political parties are now a custom normality in nearly every country, although to varying extents of influence (Renton, 1999). In more recent times, neo-fascism has been attributed largely with its online culture and presence, typified by the growing notoriety of the far-right nationalist movement the Alt-right.

4.2 Defining Fascist speech

"*Accurate text classification requires clear, consistent definitions of the type of speech to be identified*" (Durate et al., 2018).

The first hurdle to overcome when defining fascist speech is the lack of a universal consensus regarding a clear definition for it. Different commentators tend to identify different features in accordance to what they believe personify fascisms primary traits (Jones, 2014). To this end, we must accept that no matter how accurately we try to define and detect fascist speech, subjectivity will play a role in doing so. Nevertheless, we can ensure that we identify the most common and widely accepted of these features. To pursue this, we look to several historical and political commentators' assessment of the fascist philosophy.

After reviewing a wide-range of significant fascist literature, Jones (2014) stated he was 'struck' by how a select number of features seemed uniformly interleaved with fascism. He listed these to be:

- *"the glorification of the state".*
- *"the glorification of violence; and,*
- *the glorification of the leader and leadership".*

It is indisputable that fascism is more complex in essence than just this definition. Albeit, the concise three statements above appear to epitomize the vast majority of definitions given by leading sources on the matter. Acknowledged historian of fascism Stanley G. Payne provides a more comprehensive typology in his assessment of the key fascist principles (Payne, 1980). Among these he lists 'The Fascist Negations' as; anti-liberalism; anti-communism and anti-conservatism. He further identifies the fascist ideology and style to encompass:

- "*Creation of a new nationalist authoritarian state*".
- "*The goal of empire or a radical change in the nation's relationship with other powers*."
- "*Positive evaluation and use of, or willingness to use, violence".*
- "*Extreme stress on the masculine principle and male dominance, while espousing the organic view of society*".

Detailed below, we summarize some further definitions of fascism from a selection of prominent sources. By doing so, we hope to identify repeated attributes that we can use in our working definition of fascist speech.

1. Rodger Griffin: "*fascism is a term for a singularly protean genus of modern politics inspired by the conviction that a process of national rebirth (palingenesis) has become essential to bring to an end a protracted period of social and cultural decadence, and expressing itself ideologically in a revolutionary form of integral nationalism (ultra-nationalism)* ". (Griffin, 1991)

2. Kevin Passmore: "*Fascism is a set of ideologies and practises that seeks to place the nation, defined in exclusive biological, cultural and/or historical terms, above all other sources… to create a mobilized national community… it entails implacable hostility to socialism and feminism… all aspects of fascist policy are suffused with ultranationalism*". (Passmore, 2014)

3. Willie Thompson: "*There was no single defining characteristic of fascism - however, pseudo-revolutionary, authoritarian, populist, ultra-nationalism with militarised connotations covers nearly all".* (Thompson, 2011)

4. European Parliaments 1985 Committee of Inquiry into the Rise of Fascism and Racism in Europe: *"movements of the extreme right: virulent nationalism, outright rejection of democracy and of the traditional political forces and trade unionism, xenophobia, racial superiority arbitrarily defined, anti-egalitarianism, the cult of a leader, falsification of history, the glorification of certain dictatorships".* (European Parliament, 1985)

Attribute	**Definition**
Ultra-nationalism	"*Extreme nationalism that promotes the interests of one state or people above all others*".
Authoritarianism	"T*he belief that people must obey completely and not be allowed freedom to act as they wish*".
Totalitarianism	"*Centralized control by an autocratic authority*".
Supremacism	"*Advocacy of, or belief in, the supremacy of a particular group, esp. one defined by race, religion, or sex*".
Racism	"*A belief that one's own racial or ethnic group is superior, or that other such groups represent a threat to one's cultural identity, racial integrity, or economic well-being; (also) a belief that the members of different racial or ethnic groups possess specific characteristics, abilities, or qualities, which can be compared and evaluated*".
Glorification of a leader, dictatorship and or violence	"*The ascription of glory or praise to a person or thing*".
Anti-democratic	"*One who is opposed to democracy*".
Anti-feminism	"*One opposed to women or to feminism; a person who is hostile to sexual equality or to the advocacy of women's rights*".
Xenophobia	"*A deep antipathy for foreigners*".
Nativism	*"The attitude, practise, or policy of protecting the interests of native-born or existing inhabitants against those of immigrants".*
Imperialism	*"An imperial system of government; rule by an emperor or supreme ruler, esp. when despotic or tyrannical".*
Fascist	*"A member of a political organization advocating fascism; (more generally) an advocate or follower of fascism, its principles, or ideology; a proponent of any extreme right-wing ideology".*
Anti-Semitism	*"Prejudice, hostility, or discrimination towards Jewish people on religious, cultural, or ethnic grounds".*
Nazism	*"The political doctrines evolved and implemented by Adolf Hitler and his followers, esp. those asserting Aryan racial superiority, or promoting totalitarianism and the expansion of the German state".*

Table 6: The defining attributes of fascist speech. All definitions are sourced from the Oxford English and Merriam-Webster Dictionaries (2020).

As visible by the wide-range of interpretations above, fascism as an entity, appears to have a vast number of ideals associated with it. Therefore, we deem it impractical to condense our definition of fascist speech into any single description, as this would be unsuitable to incorporate the multiple traits identified. Instead, we propose a set of defining attributes that we consider to capture contemporary fascist speech. These can be viewed in table 6. For the scope of this research, we assert that if a given

text document contains language associated with any one of these attributes, then we shall signal it as fascist speech.

4.3 Distinctions vs. Hate speech

One of our ambitions in this research is to determine to what extent fascist language is intertwined with hate speech. As such, it is imperative that we also have a suitable understanding of what constitutes hate speech. Thankfully, unlike fascist speech, hate speech has a more widely accepted definition. As this study is focusing on text documents written within social networks, we can look directly to two of the leading social networking platforms to obtain their insight as to what constitutes hate speech. These definitions are displayed in table 7 and we use them as our outline for hate speech in this research.

Source	**Definition**
Facebook (Section 12 of Community Standards, 2020)	"*We define hate speech as a direct attack on people based on what we call protected characteristics - race, ethnicity, national origin, religious affiliation, sexual orientation, caste, sex gender, gender identity and serious disease or disability."*
Twitter (Hateful conduct Policy, 2020)	"*You may not promote violence against or directly attack or threaten other people on the basis of race, ethnicity, national origin, caste, sexual orientation, gender, gender identity, religious affiliation, age, disability, or serious disease."*

Table 7: Definitions of hate speech.

When considering the definitions provided for both fascist and hate speech, we can affirm that there seems to be several fascist attributes that are exclusive to fascist speech but not hate speech. For instance, attributes such as authoritarianism, totalitarianism and glorification of a leader appear to be distinctly related to fascism. As such, text documents relating to these principles will most likely be presumed as strictly fascist content (context dependant).

On the other hand, we observe parallels among certain features that both definitions specify. For example, Facebook's definition of hate speech ascribes hateful language as speech that is a 'direct attack' on a characteristic such as a person's 'race' and 'ethnicity'. Such speech is clearly aligned with language that would be associated with our identified fascist attributes of racism and supremacism.

Moreover, we hypothesize that several other fascist attributes are likely to intersect with hate speech. This is illustrated in figure 2. If this intersect exists between the two styles of language, we would anticipate a degree of difficulty when training a classifier to distinguish between the two categories of speech. For this reason, we must ensure that the data we source for our fascist text is not just reflective of the attributes within this intersect, but rather encompassing of all of the fascist attributes we have identified.

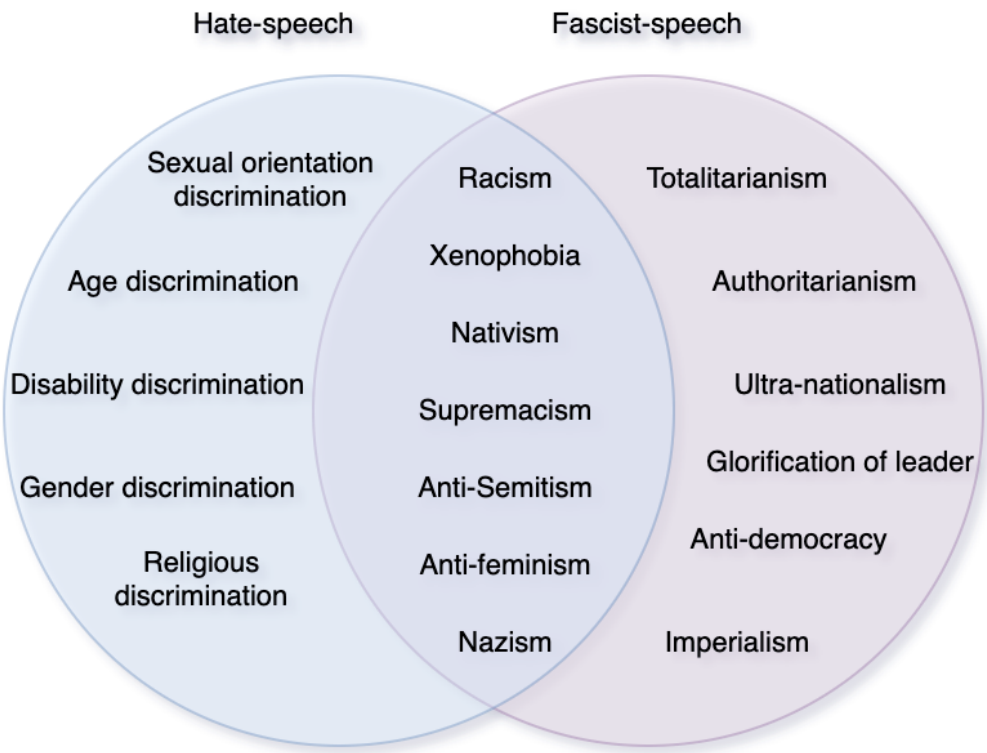

Figure 2: The intersection of fascist and hate speech.

Summary

In this chapter, we discussed fascisms significance both in a historical and contemporary context. With this knowledge in mind, and by consulting the work of some prominent researchers on the subject area, we were able to identify a set of attributes that comprise our own definition of fascist speech for use in this research (see table 6). The final subchapter provided us with a definition for hate speech, and drew to attention the possibility of a potential overlap between the two categories of speech. It is the aim of chapter 8 to investigate this possibility further, as it attempts to use machine learning methods to ascertain how distinctive fascist speech truly is from hate speech.

CHAPTER 5: Dataset Selection

Introduction

This chapter focuses on the data and datasets that were used in this study. We first touch upon the difficulties encountered in the retrieval of data containing contemporary fascist text (5.1) and the sources that were eventually identified (5.2). Subsequently, we discuss the datasets that were created from these sources and the techniques that were used to generate them (5.3).

5.1 Data Limitations

Our first investigation (chapter 7) concerns a binary classification task in which we will require ample data samples of both fascist and non-fascist text. Due to the understandable sensitivity of fascist content in modern society, finding contemporary samples of fascist text is not a trivial task. Popular social media networks are heavily monitored to prevent such opinions being circulated online, and further, such opinions may contain subtle nuances that make them hard to identify.

Several hate speech classification studies have used datasets that may contain samples which could be deemed fascist by our definition. For example, some hate speech studies have drawn from the *Stormfront* dataset which contains samples from a former white supremacist forum. On one hand this data could prove useful for some samples, as 'supremacism' and 'racism' are attributes we have identified as fascist-speech. Nevertheless, if we are not careful, too many of such samples could end up training a model that would classify racist rather than fascist speech. Therefore, samples of fascist speech will be required that represent it as an entity on a wide spectrum.

The most direct way to find wide-ranging samples of fascist speech is to search social networking sites with forums designed for the purpose of discussing and promoting fascist ideology. A selection of the most notorious websites identified containing such content include; *RedIce*; *Vanguard News Network; IronMarch*; and *The Daily Stormer.* One apparent obstacle is that such websites strive to keep their content hidden and often limit what messages are publicly viewable to non-members. In addition, many of these networks have been reprimanded for their content by having their domains removed, with some even facing legal battles (BBC, 2019).

5.2 Data Collection

Fortunately for this research, a suitable source of data for fascist text samples was identified. After going offline in 2017, data from the *Iron March* website, including all of the forums message posts was hacked and leaked in November 2019. *Iron Mach* was a neo-fascist, neo-Nazi transnational social networking community formed in 2011 by a Russian nationalist. It held ideological conversations of an extremist nature and was linked to several openly neo-fascist organizations (The Guardian, 2019). The *Iron March: Fascist Social Network* dataset was uploaded to the Data Science website Kaggle (Pham, 2019)[1], from which, we derive data from a csv file containing the networks core message posts. The file contained documents composed in various languages which we subsequently filter to those only in English, this amounted to roughly 20,000 messages.

[1] https://www.kaggle.com/davidpham/iron-march-fascist-social-network-dataset

As the samples from the *Iron March* dataset were in the form of a message forum post style, it was deemed necessary to select the non-fascist text from a source that prescribed to a similar format. We identify Reddit as a social networking site that meets this requirement. This is on the grounds that messages posted to subreddits (Reddit forums) also apply an alike style of variable message length, where messages can range from several words to paragraphs. Accordingly, a contemporary *Reddit* dataset was identified (Magnan, 2019)[2]. Also found on Kaggle, the dataset contains message posts from the 40 most popular Subreddits as of May 2019, and hence encompasses a mixed range of message discussion genres. Examples of the subreddits constituting the *Reddit* dataset include; *Funny*; *Game of Thrones*; *Gaming*; *Marvel Studios; Mortal Kombat*; *Movies*; *NBA*; *News*; *Politics*; *Relationship Advice*; *Shower Thoughts*; *Soccer*; *Today I Learned*; *Unpopular Opinions*; *Wall Street Bets* and *World News*.

An important note to make is that we make the assumption that all messages contained in the *Reddit* dataset are strictly of a non-fascist sentiment. This is due to the volume of data samples that would require to be inspected. Reddit does operate automated moderation techniques to combat hateful, and potentially fascist speech, in addition to employing manual moderators. However, as recently as June 2020, Reddit was forced down to shut a popular subreddit (contained in the *Reddit* dataset) named, 'r/The_Donald', due to its failure to control posts that promoted the glorification of violence, anti-Semitism and racism (The Washington Post, 2020). As such, message posts from this particular subreddit were discarded.

5.3 Dataset Creation

In this subchapter we present the different datasets that were produced for classification purposes. As it is not specifically mentioned below, it is important to clarify that for each dataset, we use documents selected at random from the *Reddit* dataset (see 5.2) to create the 'Non-fascist' class.

5.3.1 The Gold Dataset

We have previously mentioned that we are making the assumption that all of the *Reddit* dataset contains non-fascist samples. On the contrary however, we cannot assume that all of the message posts in the *Iron March* dataset contain explicitly fascist content. Although its discussions were held as a means to convey fascist philosophy, it is logical to assume that like the majority of online forums, some of its posts may just contain general 'chit-chat' amongst members. Consequently, if we want to create a dataset that contains truly accurate samples of our defining fascist attributes, then the only medium to achieve this is to manually annotate and label the data ourselves.

Using the attributes and their definitions provided in table 6 as a guideline, we analyse the *Iron March* dataset to extract documents identified as containing fascist speech. Roughly half a thousand samples were identified that met the required criteria. This was somewhat disappointing, as given the volume of posts contained in the *Iron March* dataset it was hoped that more fascist samples would be derived. Nevertheless, 500 fascist documents provided us with options to work with. Through the manual annotation process, we create a 'Gold' standard dataset. The term 'Gold standard' originated from the economical monetary system but has since been adapted to refer to scientific procedures and collections which conform to a set standard (Almashraee et al., 2014). We therefore refer to this dataset as the *Gold* dataset. From the best of our knowledge, this is the first dataset created containing contemporary annotated samples of an explicit fascist nature.

[2] https://www.kaggle.com/smagnan/1-million-reddit-comments-from-40-subreddits

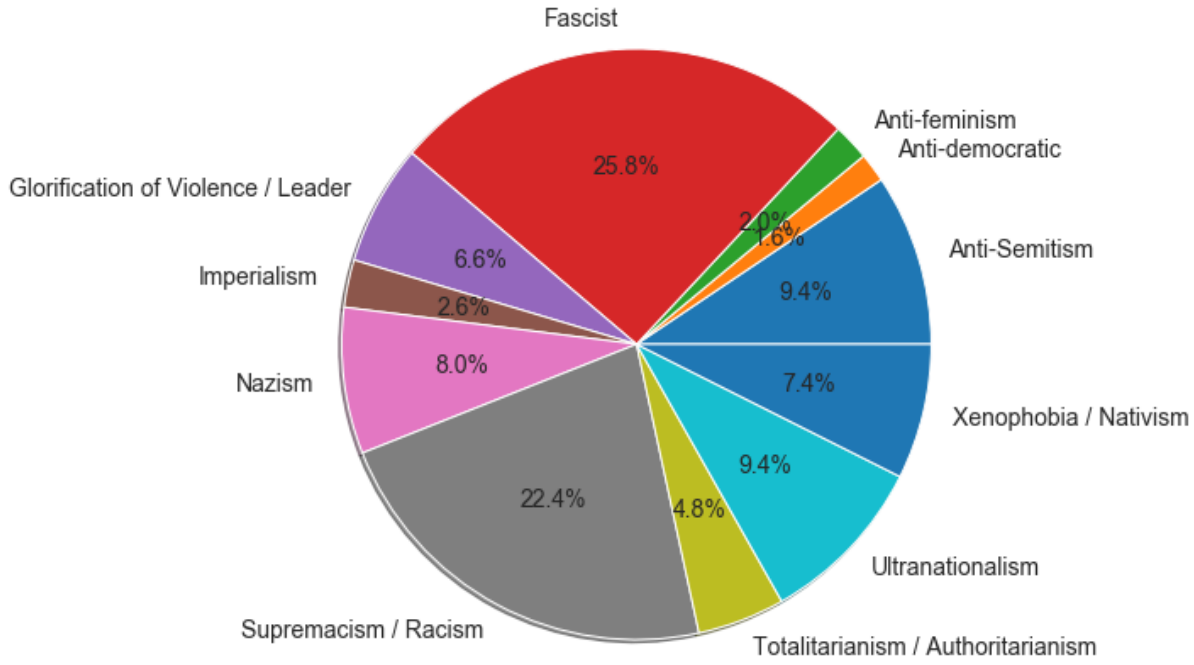

Figure 3: Attribute distribution of fascist samples in Gold dataset

Like Warner and Hirschberg (2012), we also assigned fascist samples to one of several categories derived from our attribute definitions. Unlike them however, we did not assign samples to multiple attributes. For this reason, certain fascist samples may contain language associated with several of the fascist attributes, and yet, they have only been assigned to the most prevailing one. In addition, as it became promptly noticeable that certain attributes, such as 'supremacism' and 'racism' were greatly intertwined, we combined them together into the same category. Through this process, we were able to create a graphic representing the attribute distribution of the fascist data we had extracted. This is illustrated in figure 3. This process was considered integral in confirming that we had gathered data reflective of our definition of fascism in chapter 4.

Although we had created a suitable dataset for use in our investigations, it was further deemed necessary to create datasets of a larger size. This is due to the knowledge that text classification models typically perform better and are more robust to real world data when they have an abundant number of samples to train on. More data instances allow us to tackle more ambitious tasks and are crucial to achieving good generalization (Domingos, 2012). In addition, small sized datasets can result in a model overfitting. Therefore, to increase the chances of success, alternative techniques were considered to expand upon this data.

5.3.2 The Shuffled Dataset

When posed with a shortage of data, one solution is to consider data augmentation techniques to generate new samples for minority classes (see 2.1.3). One augmentation approach for text data that was identified is the process of sentence reordering. Like the name suggests, sentence reordering is the process of shuffling the order of sentences within a given text document to form new samples. The method we draw upon to augment textual documents via sentence reordering was derived from a publication by Tjihero (2018).

It follows the logic, for a given input sequence of text where:

X = A single text document.
x_1, x_2, x_3, x_4 = Individual sentences that comprise the given document.

Take the document:

$$X = x_1, x_2, x_3, x_4$$

Tokenize (see 6.1.1) its sentences and shuffle them at random:

$$X = x_1, x_3, x_4, x_2$$

If the newly formed document does not already exist, we can append it as a new sample in our dataset. The process if further illustrated in figure 4.

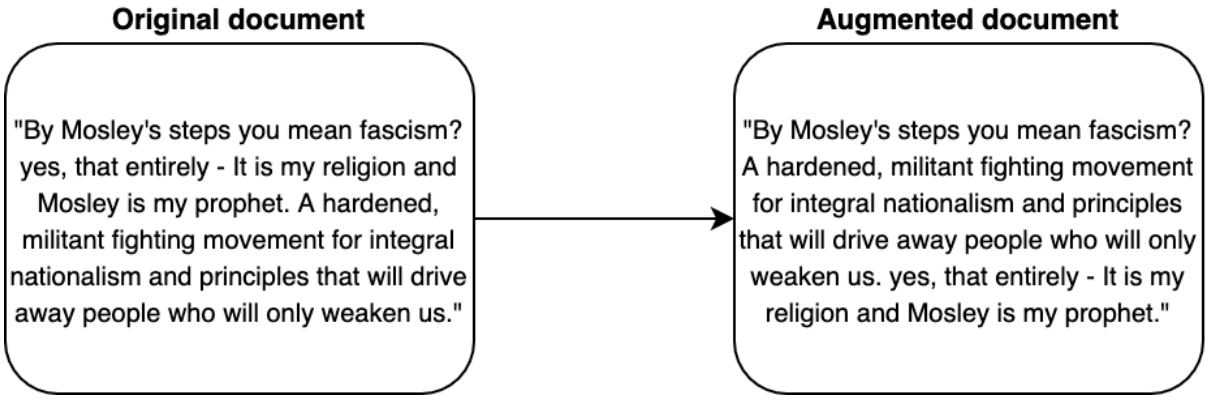

Figure 4: Sentence reordering augmentation, before and after.

As is evident, such a process can only be used on documents containing at a minimum more than one sentence. However, one notable advantage provided by sentence reordering is that the context and sentiment of the document is still preserved in the augmentation process. We shall refer to the dataset that has been augmented via sentence reordering as the *Shuffled* dataset. To create this, we apply the sentence reordering augmentation technique on the fascist samples derived directly from the *Gold* dataset.

One important finding regarding data augmentation denotes that only the training data should be augmented, and importantly, this should only take place after the data has been carefully split into training and test sets (Gröndahl et al., 2018). This is due to the complication of data 'leakage' if performed on the entire dataset, as we want to ensure that all aspects of the test data remain wholly unseen by our model. If not performed correctly, this would lead to inaccurate results. Therefore, we adhere to this guidance for the *Shuffled* and all subsequent datasets that are created by means of an augmented or synthetic procedure.

5.3.3 The SR Dataset

A further augmentation technique we employ is synonym replacement. This is the process of selecting *n* number of words from a given text document and replacing them with their synonyms. This process can be performed either at random or based on a probability or closeness threshold as noted in Zhang et al. (2015). The method of synonym replacement used in this research was adopted from Wei and Zou (2019). We utilize their publicly available code from their paper, *Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks*, to achieve this means. Like Zhang et al. (2015), it uses a WordNet lexical database of semantic relationships to identify synonym replacement options for words and selects them at random. This process is illustrated in figure 5.

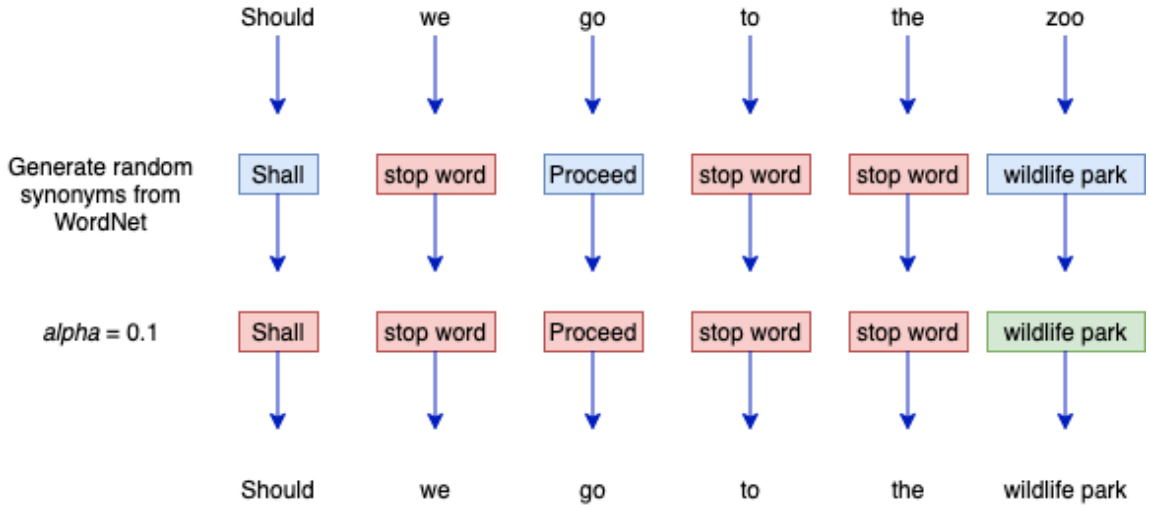

Figure 5: Synonym replacement augmentation, the process.

Unlike the previously detailed sentence reordering method, synonym replacement augmentation has the distinct advantage of creating entirely new documents of text. Crucially, we are still maintaining the original context of the documents in the process. To determine the percentage of words we want to transform we can adjust the *alpha* parameter as appropriate. If *alpha* is set to 0.4 for example, roughly 40% of the words within a sentence will be transformed. The *n_aug* (number of augmentations) parameter also allows us to specify the number of augmented sentences we wish to generate from each original sentence. To generate new fascist documents for this dataset, we set (*alpha* = 0.2) and (*n_aug* = 2). Correspondingly, we expect to generate two new documents per original with roughly 20% of the text transformed. The transformation of a fascist document by virtue of synonym replacement is portrayed in figure 6. We refer to the dataset that has been transformed by means of synonym replacement augmentation as the *SR* dataset. Analogous to the *Shuffled* dataset, the original fascist samples are taken from the *Gold* dataset.

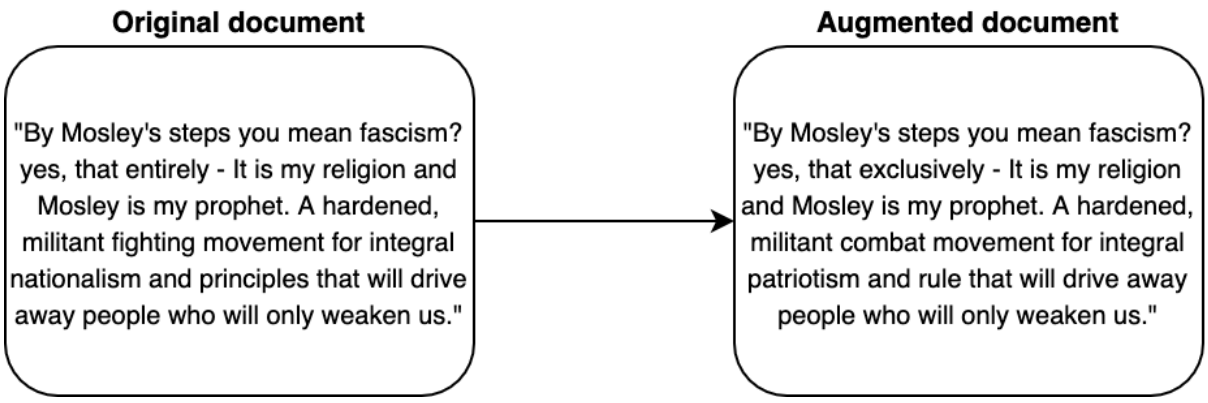

Figure 6: Synonym replacement augmentation, before and after.

5.3.4 The Synthetic Dataset

An alternative means to generate a greater number of data samples is to create artificial samples based on the distribution of existing data. One such method to achieve this is through SMOTE (Synthetic Minority Oversampling Technique) proposed by Chawla et al. (2002). Using the k-NN (k-nearest neighbours) algorithm, SMOTE performs oversampling on minority classes by creating new synthetic data instances. To form a new sample, the algorithm first selects an instance at random from the minority class and subsequently forms a line between it and a random nearest neighbour. Synthetic data is then generated between the two instances as a convex merger to form a new sample. The process is then repeated until the imbalance has been offset.

We create a *Synthetic* dataset whose additional fascist samples have been artificially generated via SMOTE. It is worth noting that SMOTE oversampling is an unconventional method to use on textual data, as more commonly it is applied to numerical oriented tasks such as fraud detection. Consequently, the oversampling is only applied on the input data after it has been transformed into numerical feature vectors. Despite its unconventionality, we highlight its success in past sentiment classification tasks (Xu et al., 2015) and (Indurthi et al. 2019) as reasoning for its inclusion in this research.

The aforementioned augmentation techniques serve to increase their respective datasets training instances, albeit, only by relatively modest numbers. Therefore, we elect to push the boundaries whilst applying SMOTE by generating thousands of artificial fascist training samples. Our motivation for this stems from our chosen feature extraction techniques (see 6.2). Two of which (word embeddings and Paragraph vector) rely on ample amounts of data to form accurate word representations. As such, it is anticipated that the vastly increased training samples will have a positive effect on these features' performance.

Summary

In this chapter we discussed some of the limitations we faced regarding the retrieval of fascist text documents and the data sources that we identified for both the fascist and non-fascist text in the form of the *Iron March* and *Reddit* datasets. The creation of a manually annotated *Gold* standard dataset provided us with high-quality data but was somewhat lacking in quantity. With the purpose of providing our models with the best chances of success in the later classification investigations, we used several augmentation techniques in addition to the SMOTE oversampling method to create three diverse and larger datasets.

CHAPTER 6: Methodology

Introduction

Chapter 6 focuses on the methodology that was executed prior to the classification investigations. We begin by discussing the data pre-processing methods that were employed on the data (6.1). Subsequently, we describe the feature extraction techniques that were selected to transform the data (6.2). The model selection process is also detailed, including which classification algorithms were selected (6.3), validation techniques (6.4) and parameter tuning (6.5). Finally, we illustrate the test system architecture that was used to run the experiments (6.6).

6.1 Data Pre-processing

After collecting our data, we must ensure it is in a clean a format as possible before we transform it into features suitable for modelling. Data pre-processing techniques helps us to retain the information within text that we consider useful, whilst also allowing us to discard irrelevant noise from our data. It is an essential phase in the NLP pipeline. Before being used to train and fit a model, the data underwent the following pre-processing steps.

6.1.1 Tokenization

Tokenization is the process of breaking down text into individual units referred to as ‘tokens’. To tokenize a string, we divide it into linguistical components that comprise a portion of text data. Tokens are the foundation from which we can derive the semantic meaning of language, as we will attempt to infer the relationship from one token to another. The NLTK (Natural Language Tool Kit) library in Python contains its own tokenizer that we utilize to achieve this objective. The process of converting raw text into tokens can be observed in figure 7.

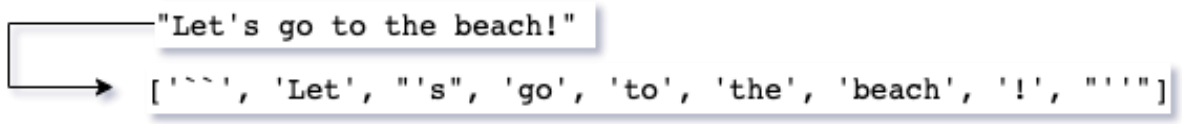

Figure 7: Tokenization of a String via NLTK TreebankWordTokenizer.

6.1.2 Stemming

Stemming is a common technique used in NLP with the aim of reducing the inflectional or derived forms of words by confining them to a root form, known as a word’s ‘stem’. This is achieved by stripping a given word of its affix, which refers to any addition to a word from its stem form. The decision of what constitutes a words stem is based upon the algorithm of the stemmer in question. For example, if we took the words; ‘connected’, ‘connection’, ‘connections’ and ‘connects’ - we may stem all of these words to the root form, ‘connect’.

We opt to employ a Snowball stemmer (Porter, 2001) to stem our raw text, which is also sourced by the NLTK library. The Snowball stemming algorithm was created by Porter to address some of the shortcomings in his original implementation of the Porter stemming algorithm, including an improvement in computational speed and logic. The Lancaster stemmer was also considered, however, being the more aggressive of the stemming algorithms and due to the desire to preserve as much lexical context as possible it was deemed unsuitable for this task.

6.1.3 Additional steps

In addition to the above, we further normalise the text data by applying the following steps:

- The removal of English stop-words (e.g. ‘is’, ‘at’, ‘for’). Stop-words represent frequently used words that provide us with little semantic meaning.
- The removal of emoticons.
- The removal of URLs.
- The conversion of all text to lower case.
- The removal of punctuation.
- The removal of irrelevant characters (e.g. ‘< ‘, ‘€’).
- The replacement of numbers with their textual representations (e.g. ‘10’ becomes ‘ten’).

6.2 Feature Extraction

Feature extraction is the process of transforming raw text data into numerical feature vectors that can be used for modelling purposes. As such, we can think of it as the numerical representation of a word that conveys a relative meaning. This process is essential to make the data interpretable for machine learning algorithms. There are a wide range of feature extraction methods that can be adopted for the task at hand (see 2.2). We narrow these options down by selecting three different feature extraction techniques, each with a contrasting style. These include TF-IDF (Term Frequency-Inverse Document Frequency), word embeddings and Paragraph vector. In this subchapter we detail their approaches and our rationale for these choices.

6.2.1 TF-IDF

TF-IDF is a statistical measure which can be used to evaluate the importance of a term in a document, given the importance of a term in an entire corpus of documents. It combines two very interesting concepts.

TF (Term Frequency): First we consider the basic count of how many times a given term occurs in a given document and divide it by the total number of documents that contain that term:

$$tf(t,d) = \frac{f_d(t)}{n_d} \tag{6.1}$$

Where $f_d(t)$ is the frequency of term t in document d and n_d is the total number of words in document d (Portilla, 2019).

Through this means, words that appear more frequently in a given document are assigned more weight than those appearing less frequently. This alone may not be adequate to capture the meaning of a document, as words that appear less frequently that still hold a significant meaning could be penalized. In addition, with this approach commonly used terms with little meaning would yield higher scores. Even with the highest prevailing stop-words removed there would likely remain certain words that arise consistently that are non-specific to the document or corpus.

IDF (Inverse Document Frequency): Subsequently, we seek to address this issue by increasing the weight of words that occur more rarely whilst decreasing the weight held by terms that appear more frequently within the entire corpus. Therefore, IDF allows us to gain the importance held by a term:

$$idf(t, D) = \log \frac{\mathrm{D}}{\{\mathrm{d} \in \mathrm{D}: \mathrm{t} \in \mathrm{d}\}} \tag{6.2}$$

where D is the corpus count of documents (Portilla, 2019).

TF-IDF (Term Frequency-Inverse Document Frequency): By combining these two aspects, TF-IDF seeks to consider the importance of a term appearing in a document, by offsetting it with how important that term is in relation to its weight in the entire corpus of documents:

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \tag{6.3}$$

TF-IDF was selected as a feature extraction method due to both its popularity and its success rates in text classification tasks. With the use of the Scikit-learn machine learning library, we create two TF-IDF vectorizer classes to be used as feature transformers in the machine learning pipeline. The first was designed to extract word n-grams. We set the ngram_range parameter to (n = 1, 3), meaning that we will consider word n-grams that are unigrams, bi-grams and tri-grams. The second we create to extract character n-grams, which we set in the range of (n = 2, 6).

6.2.2 Word embeddings

The second approach we use for feature extraction is to use word embeddings (see 2.2.2). This technique groups words that share certain similarities closer together within the vector space through cosine distances. As such, words that are semantically similar and used in related contexts will be represented by vectors that have a higher cosine similarity. Through this means, word embeddings seek to capture the contextual significance of words. Once an adequate number of vectors have been thoroughly learned, we can arithmetically manipulate them to discover their hidden semantic relations. A well-known example is that if we took the vector for ‘King’, subtracted ‘Man’, and added ‘Woman’, we would hopefully deduce the word vector for ‘Queen’. For classification purposes, we use the learned word associations to infer predictions on our data.

To implement this, we adopt the Word2vec algorithm proposed by Mikolov et al. (2013) which uses a two-layered shallow neural network model to learn its associations. Word2vec comes with two possible implementations, both of which are displayed in figure 8. The first, CBOW (Continuous Bag of Words) utilizes *n* number of surrounding context words to predict a word in the middle, known as the centre word. Whereas the alternative, known as a Skip-Gram, uses the centre word as a channel to learn from its surrounding words and phrases. For example, if we took the phrase, “*the quick brown fox jumps over the lazy dog*”, where our centre word = ‘jumps’ with a window size = 2. In a Skip-Gram implementation, the neural model will construct the training samples; (jumps, brown), (jumps, fox), (jumps, over) and (jumps, the). The logic behind this is that over time words such as ‘fascist’ and ‘state’ will appear in the same context more frequently than ‘fascist’ and ‘skiing’, and hence their vectors will be mapped closer together.

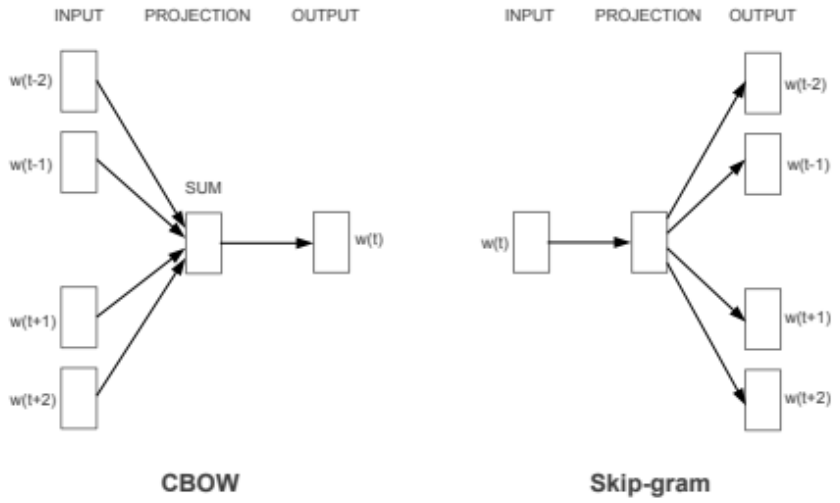

Figure 8: Word2vec architectures CBOW and Skip-gram (Mikolov et al., 2013).

Concerning our implementation of the Word2vec algorithm, we elect to use pre-trained word vectors from Google's own model (Google, 2013). These contain 300 dimensional vectors that were trained on a Google news corpus of over 3 million words and phrases. The logic behind our decision to incorporate this is due to the vast amounts of data these vectors have been provided to learn word associations from. The Skip-Gram model we incorporate was trained on roughly 100 billion English words, so we can hopefully assume that the majority of words in our datasets should have learned vectors for prior to transformation.

We create a mean word embedding class for the pre-trained vectors with its own fit and transform functions. This allows it to be used as an intermediate step in the form of a feature vectorizer in combination with the Scikit-learn Pipeline class. The KeyedVectors class from the genism library was used to load in the word vectors and the pre-trained vectors are then used to compute vector representations for each word in a given document. Thereafter, the mean value is computed to generate a single vector which is used to represent the given text document. If in the unlikely event a word in our data does not appear in the pre-trained vector vocabulary, we discard it.

To provide a visual representation of the capability of word embeddings, we use the t-SNE tool from the Scikit-learn library to allow us to visualize data of high-dimensionality. Figure 9 illustrates a Word2vec networks vector representation of words from our fascist data established from their semantic similarities.

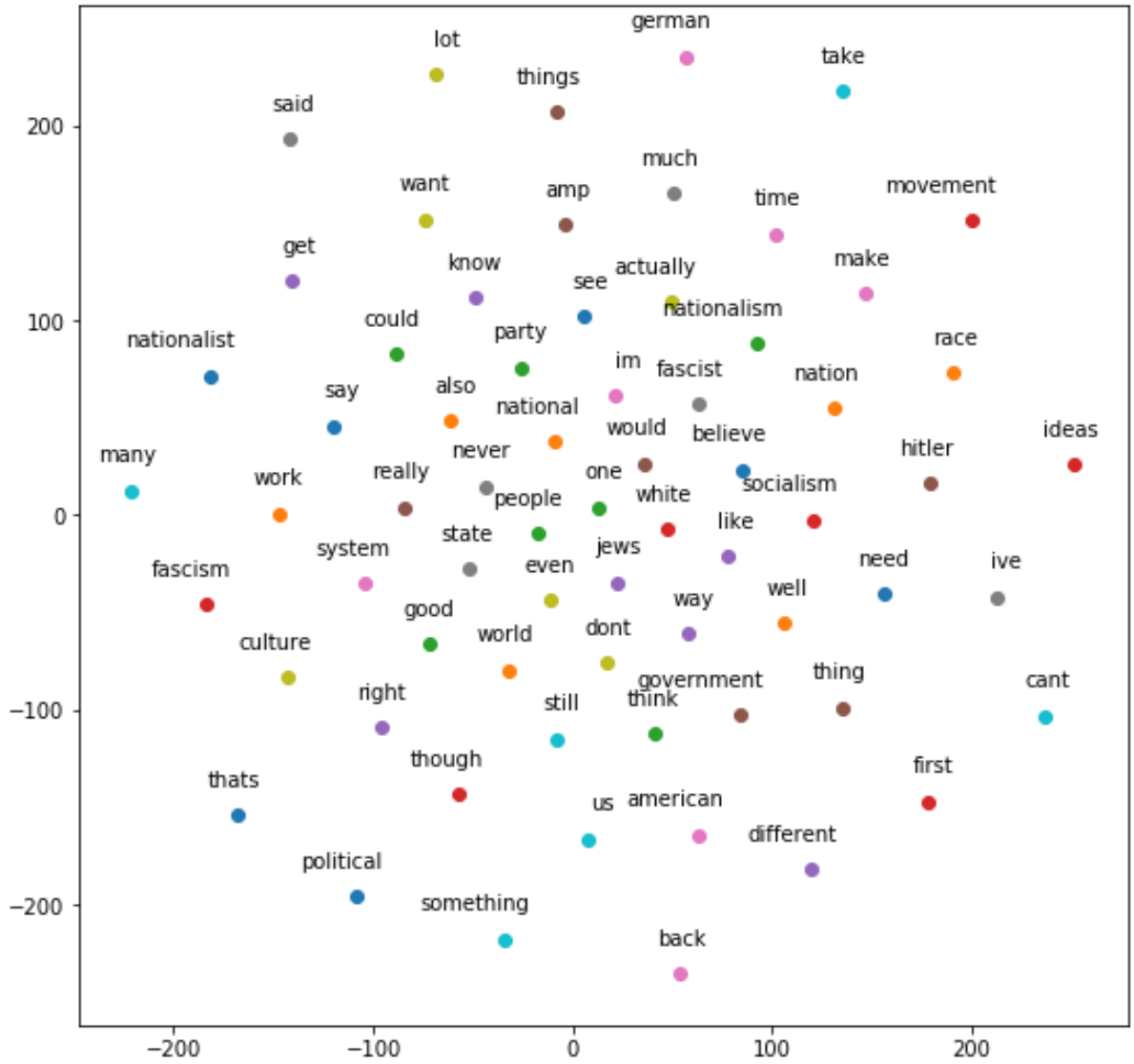

Figure 9: 2D representation of word vector space produced by Word2vec network.

6.2.3 Paragraph Vector

Whereas word-level embeddings generate a feature vector for each word within a given document, alternative embedding approaches are designed with the intention of achieving phrase, sentence and even paragraph-level representations. Paragraph vector, proposed by Le and Mikolov (2014), is capable of computing representations for text sequences of a variable length. Consequently, it can capture vector representations for a document of any size or shape. In literature it is often referenced as Doc2vec or Paragraph2vec, and sharing the same creator as the Word2vec algorithm, it adheres to a similar logic.

Paragraph Vector shares a resemblance with word embeddings in the manner that it computes vectors for words within a document. Its key distinction however, is that it further computes a paragraph vector to represent the context of the document itself. Subsequently, it concatenates the word and paragraph vectors together to use in union to infer predictions. The rationale behind this is that a feature vector representing the given document should aid when predicting or classifying the words within it. The distinction between the Paragraph vector and Word2vec framework is depicted in figure 10. The addition of the paragraph vector in document matrix *D* is to represent the context

and overarching theme of the remaining words in the document. When concatenated with the three context words from a word matrix *W*, a decision can be inferred to predict the fourth word.

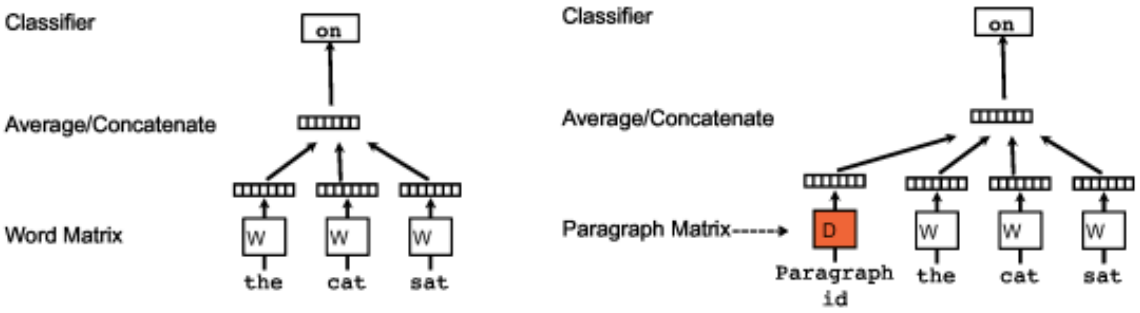

Figure 10: Word2vec framework for learning word vectors (left) and Paragraph vector framework for learning paragraph vectors (right) (Le and Mikolov, 2014).

The appeal of Paragraph Vectors ability to contextualize entire documents is evident. We demonstrate this with an example: Imagine if our corpus contained multiple occurrences of the phrase, "shuffle the deck", and we were subsequently tasked with having to predict what followed a new occurrence of the first two words ("shuffle the") in a new document. The chances would imply that the next word would indeed be "deck", however, now consider that the subject of this document is actually Computer Science. Hopefully, Paragraph vector may have inferred the subject theme from the document, and rather than predicting the word "deck", it would predict words such as "array" or "list" instead.

Like Word2vec, Paragraph Vector comes with two possible implementations. PV-DM (Distributed Memory) functions in the exact manner as described above and observed in figure 10. A paragraph vector is used in combination with a number of context words to determine predictions. Conversely, the PV-DBOW (Distributed Bag-of-Words) model does not utilize the context word vectors, instead relying on only the paragraph vector itself as an input to generate predictions for a window of context words. This is conveyed in figure 11.

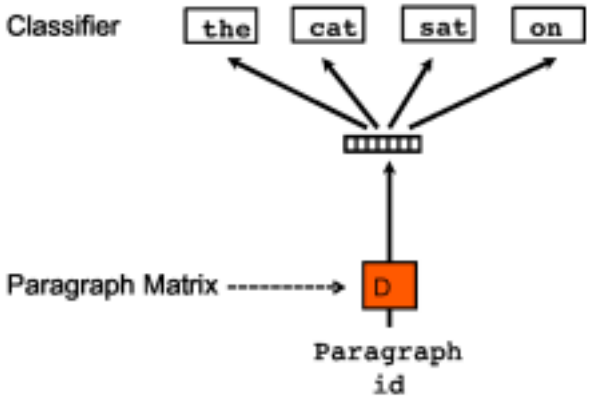

Figure 11: PV-DBOW Paragraph vector model (Le and Mikolov, 2014).

Paragraph Vector uses a neural network to learn its vector representations, the word and paragraph vectors are trained via stochastic gradient descent which is attained by means of backpropagation. In contrast to the word vectors, the paragraph vector itself is computed only at time of prediction as an additional stage. To compute a paragraph vector for *N* paragraphs in a corpus and *M* words within its vocabulary, every paragraph gets mapped to *p* dimensions and every word is mapped to *q* dimensions. Resultantly, the model obtains $N * p + M * q$ parameters to learn its paragraph vectors from (Le and Mikolov, 2014).

Contrary to our approach with word embeddings, we train the Paragraph vector network using the vocabulary within our own data. Although this is considerably less data, one benefit of using our own data is that the vectors generated should be more specific to the fascist language domain. Nevertheless, the impact that the sparsity of data has on the network's ability to generate suitable feature vectors will be something to monitor closely in the investigation results.

For the implementation of the Paragraph vector transformer we again utilize the genism library. The infer_vector function is applied to compute feature vectors for words and documents. In addition, the TaggedDocument function is used to assign each document a unique ID so it can be used as an input in the paragraph matrix *D*. Our custom Paragraph vector class adopts the PV-DM architecture, training for a total of 20 epochs. In accordance with our previous features, we create the Paragraph vector class with transformer functions. Several combinations of the network's parameters were tested in an attempt to determine the most suitable settings.

6.3. Model Implementation

After we have transformed our raw text into numeric feature vectors, we must select classification algorithms to interpret the data and build suitable models from. This subchapter concerns the classification algorithm selection process that was carried out in this research. Three machine learning algorithms were identified to trial and evaluate in total. All algorithms were implemented using the Scikit-learn library in Python.

6.3.1 Support-Vector Machine

Support-Vector Machines (SVMs) are supervised learning algorithms which can be used in classification (linear and non-linear), regression and outlier detection tasks. SVMs focus upon the extreme data points, referred to as 'support vectors', among classes. The support vectors are used to push up against the geometric margin of the classifier such that the most optimal position to set a separating line between classes is found. Once established, the algorithm formulates a decision boundary, known as a 'hyperplane', that acts as the line to segregate the data and to maximize the margin (Kotsiantis, 2007). This process is illustrated in figure 12.

We select SVM as our first model due to both its popularity and its suitability for classification tasks with small to medium sized datasets (Gèron, 2019). We implement our version of the SVM as a Linear-SVC (Support-Vector Classifier), as these models have been proven to perform well on a wide variety of text classification tasks, including Warner and Hirschberg (2012). The linear kernel function specifies that the problem to solve should be linearly separable, meaning, each class can be distinctly separated from one another by a hyperplane.

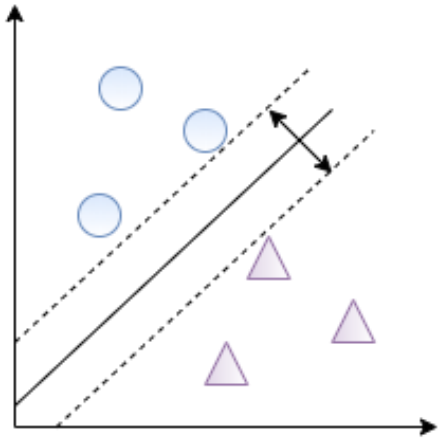

Figure 12: SVM model setting hyperplane between two linearly separable classes.

6.3.2 Logistic Regression

Logistic Regression is a popular statistical model that is commonly used as a classification algorithm. It computes the weighted sum of its input features (combined with a bias) using a non-linear function called the logistic function (also known as the sigmoid function) which transforms its output to a given value between two horizontal asymptotes ranging from 0 to 1 (Molnar, 2020). The hypothesis expectation of a logistic regression model is as follows:

$$0 \leq h_\theta(x) \leq 1 \tag{6.4}$$

Where the hypothesis h_θ of input x is to be transformed to an output between 0 and 1 by the sigmoid function g:

$$g(x) = \frac{1}{1 + e^{-x}} \tag{6.5}$$

After the model has determined the probability p of input x being attributed to the positive class, it can then make its prediction $\hat{y}$ (Gèron, 2019):

$$\hat{y} = \begin{cases} 0 \; if \; \hat{p} < 0.5 \\ 1 \; if \; \hat{p} \geq 0.5 \end{cases} \tag{6.6}$$

When expanded from binary to multiclass classification, we refer to Logistic Regression as Multinomial Regression (Molnar, 2020).

The overall aim of logistic regression is to find the best fit line, known as the decision boundary, which can be used to predict future classes. It uses a cost (or loss) function when training to penalize the model for misclassification errors, such that the parameters are set for the model to predict high probabilities for instances that are positive and vice-versa for the negative instances (Gèron, 2019). Gradient descent is an optimization algorithm often used in logistic regression to minimize the cost function. It calculates the error for each prediction within the training set and calculates new coefficient values based on any errors in the prediction. This is repeated until the algorithm has found the global minimum (Gèron, 2019).

We select Logistic Regression for its proven success in binary text classification tasks. Moreover, we look to the notable performance of the Logistic Regression model produced by Davidson et al. (2017) in their automated hate-speech study as a benchmark for its potential capability in this field of classification.

6.3.3 Random Forest

Random Forests are ensembled learning methods which can be used in both classification and regression tasks. They construct a number of Decision Trees at the time of training which they use to form predictions. Decision Tree models take in an input vector of attribute values and return an output value in the form of a decision. The output decision is reached by the tree performing a sequence of tests which are passed through the its structure. Each node in the tree represents a test for a given attribute and each branch represent a possible outcome of the test (Kotsiantis, 2007). They are considered a rule-based approach to classification, as they employ conditional branching (if/else statements) as a series of decisions to be executed to reach a prediction.

When populated to infer decisions by a Random Forest, each individual Decision Tree makes its own prediction on an input instance which is aggregated alongside the other tree's predictions. If adopted as a classification algorithm, the mode of the decisions is taken forward as the output class prediction for the given instance (James et al., 2017). Decision Trees can adopt one of several algorithms to help them make decisions. The algorithm selected will inform the trees how to split their nodes based on the target variable. One commonly used for classification is the Gini Index, which acts as a cost function to inform split decisions based on node 'purity':

$$G = -\sum_{k=1}^{K} \hat{p}_{mk} \, (1 - \hat{p}_{mk}) \tag{6.7}$$

Where $\hat{p}_{mk}$ denotes the number of training observations in the m^{th} region from the k^{th} class (James et al., 2017).

Our reasoning to elect Random Forest as our final algorithm was twofold. Its use of 'bagging' (bootstrap aggregating) makes it less-prone to overfitting than the sole deployment of a Decision Tree, as this enables a reduction in variance by evening-out decisions (Burnap and Williams, 2015). Moreover, Random Forests have also been proven to perform and contend well with large datasets containing higher dimensionality (James et al., 2017). Therefore, it will be interesting to see how this approach performs with the large volume of training documents produced by the SMOTE oversampling technique within the *Synthetic* dataset in comparison to the others.

6.4 Validation Techniques

In machine learning, validation concerns the way in which we evaluate a trained model with previously unsees test data. Hence, validation techniques refer to the methods used to separate data into training and test sets. During the classification experiments that were carried out in chapters 7 and 8, two contrasting validation methods were used. The rationale for the different approaches is covered in the subchapters below.

6.4.1 Cross-validation

The first validation technique we adopt is k-fold cross-validation. Cross-validation is distinct from other validation techniques in that it uses all the available data to both train and evaluate a model. When applying this technique, we specify *k* number of folds we wish to split our data into. If $k = 5$, the model will separately train and be evaluated five different times. Upon each iteration, the model trains on four of the folds reserving one for testing purposes. Once the iteration has finished, the fold that was last used for testing purposes now becomes training data, and so forth, until each fold has been utilized as a test set. This process is depicted in figure 13 (see 6.5). Upon completion, the mean output scores of each test fold is computed to ascertain a model's performance metrics. To perform cross-validation we implement a stratified 5-fold cross-validation. By opting for stratified folds, we ensure that each fold has roughly the same proportion of data instances for each class.

For the reason of preventing data leakage between training and test sets, this validation technique could only be used on the non-augmented datasets. Namely, the *Gold* dataset. As the augmented datasets contained several modified versions of original samples, it was critical to make sure that both the original and augmented samples remained in the same train-test set. If not adhered to, when evaluating a model with test data, it is likely that the model will have already seen many similar augmented versions of the same instance whilst training, and thus, would lead to inaccurate results. Therefore, for the *Shuffled*, *SR* and *Synthetic* datasets, we instead use the hold-out validation technique (see 6.4.2).

6.4.2 Hold-out (train-test split)

The second technique used to formulate train and test data is referred to as 'hold-out', also known as the train-test split. In this process, data from each class is randomly separated into train and test sets based on a designated percentage. The model trains exclusively on the training data and is later evaluated based on the predictions it makes on the unseen test data. One advantage of such a process is that it is relatively easy to grasp and perform. The downside conversely, is that not all of the data is being utilized for both training and test purposes. Thankfully, we are only using this technique on the larger augmented datasets and so this doesn't induce great concern.

To perform the train-test split we use Scikit-learns train_test_split function. This allows us to specify the quantity of data we wish to hold-out for testing whilst also dividing instances at random. When partitioning the augmented datasets, we opt for a standard practice split in the ratio of 70:30 in favour of the training data.

6.5 Grid Search Hyperparameter Optimization

The optimization of a machine learning models hyperparameters is an important process, as it can have a notable impact upon its performance. Hyperparameters are parameters that cannot be learned directly from the regular training process, such as the number of trees in a Random Forest model for example. Due to the computational cost and time required to test the numerous settings of each model, we instead opt to conduct a grid search to tune our models hyperparameters by means of cross-validation. The grid search technique evaluates each combination of a models hyperparameters that we specify and subsequently sets the values to those deemed to be most optimal. The process of conducting the grid-search is illustrated in figure 13. As this technique is required to be carried out via cross-validation, again, we can only apply this to the non-augmented datasets (see 6.4.1). Therefore, models utilizing the *Gold* dataset underwent a grid search for hyperparameter tuning, whereas we

leave those models used in combination with the *Shuffled*, *SR* and *Synthetic* datasets with their default settings.

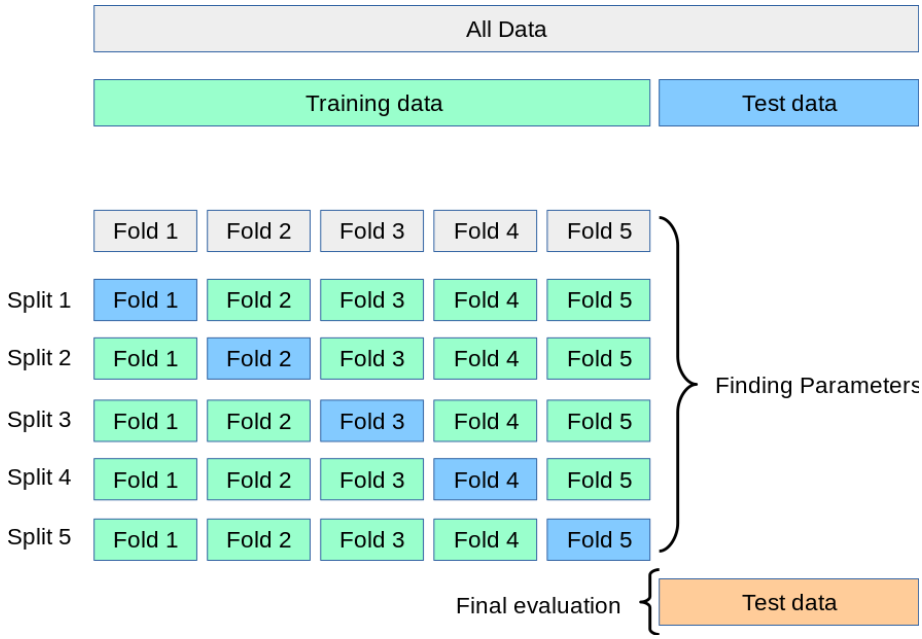

Figure 13: Grid search cross-validation (Scikit-learn.org, 2020)

6.6 Test System Architecture

All of the code designed for classification purposes was written in Python files (version 3.8) in PyCharm, the only exception to this was the initial dataset creation for which a Jupyter Notebook was used. Python was chosen as the preferred programming language due to its connection with highly-regarded machine learning and NLP libraries. Java was also considered for its familiarity and access to the Weka API. Since Python was an unaccustomed language, an initial period of time was devoted to learning its basic syntax and becoming familiar with the relevant tools and packages.

To perform the experiments that are detailed in chapters 7 and 8, a test system was designed to implement the classification functionality. The principle of the system is to allow the user to input their desired feature extraction technique and machine learning algorithm as parameters, after which models for the applicable datasets shall be constructed, trained and evaluated. Subsequently, the performance metrics and confusion matrices (see 3.5.1) are printed for the user's assessment. A component diagram depicting the basic layout of the system is illustrated in figure 14.

The Classification component contains the Python file 'classification_experiments.py' which holds the two methods used for classification. The first, make_classification(), is used for the binary classification experiments and conducts the tests for all four datasets concurrently. In addition to allowing the user to decide between algorithm and feature selection, it takes grid_search_tuning as a third Boolean parameter, which enables the user to choose whether a grid search takes place for hyperparameter optimization. Conversely, the second method, make_multi_classification(), accomplishes a similar task, however this time regarding multiclass classification.

The remaining components carry out the additional functionality that are needed to perform the classification tests. Most of the pre-processing and data cleaning functions were applied to the text

data upon the creation of the original datasets. The stemming process and the removal of stop-words, however, do not occur until the datasets are loaded into the system, and these functions are imported form the Pre-processing component. The Utils component contains the majority of functions and common constants that are used to build the classification pipelines. This component also acquires the user's choice of model and passes it to the grid search for tuning. Finally, we create a Data Visualisation component that was used for data exploration and data analysis purposes.

During this study we utilize several of Python's libraries to help assist us. As previously mentioned, Scikit-learn hosts a wide range of open sourced machine learning utility interfaces. It was used to implement the machine learning algorithms and to fit the classification pipelines. The NLTK library provides fantastic packages to aid with language processing, and it was from this resource we acquire our Snowball stemmer and list of English stop-words. Moreover, the Imbalanced-Learn library was utilized for the SMOTE oversampling of the minority training classes in the synthetic dataset. Concerning data visualisation, both the Matplotlib and Seaborn libraries provide high-level interfaces for designing appealing statistical graphics. For this reason, they were selected to design the confusion matrices and additional graphics used throughout this research.

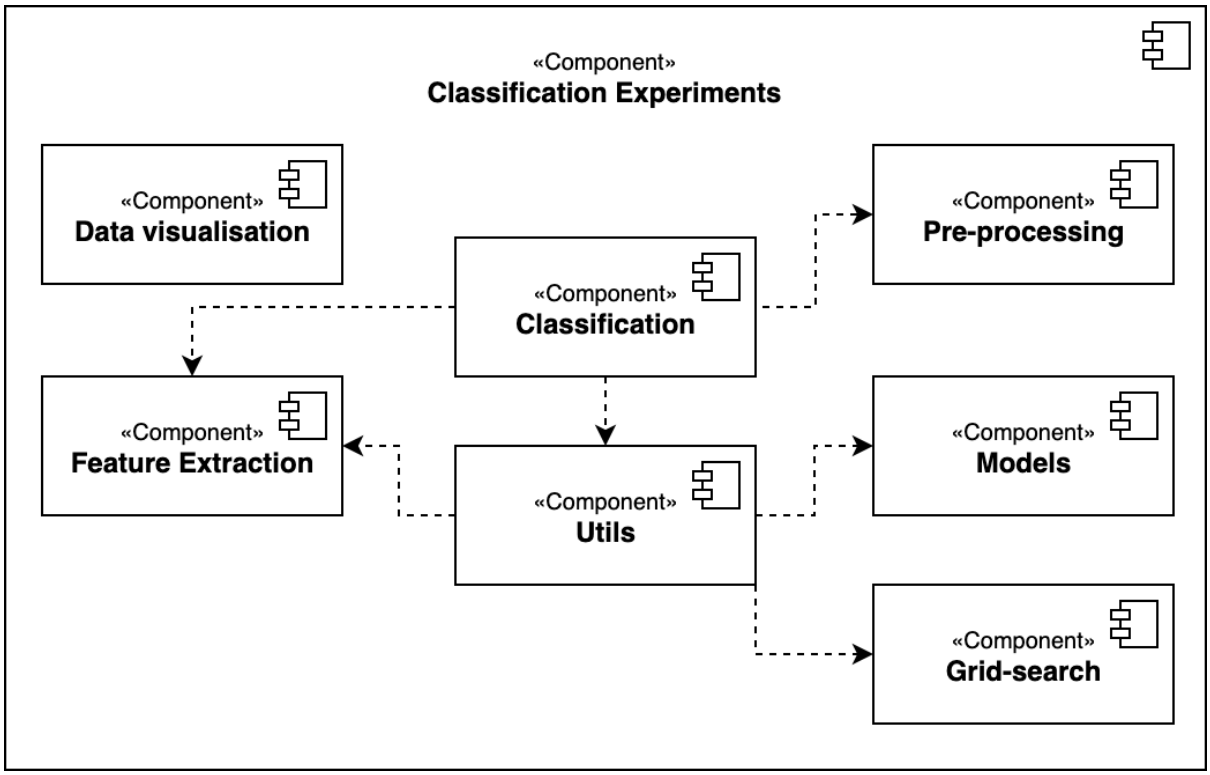

Figure 14: Component diagram of test system architecture.

CHAPTER 7: Binary Classification (Fascist vs. non-fascist speech)

Introduction

This chapter conveys the results and provides an evaluation of the experiments that were conducted concerning binary, fascist vs. non-fascist classification. The motivation for this investigation is to first establish whether we can successfully train a classifier to recognize fascist text. In addition, in the process of reaching this verdict, we also want to evaluate and compare a range of machine learning algorithms and feature extraction techniques in their abilities to classify fascist text and surmise the affect that the size and quality of dataset has on the results.

7.1 Data Description

All datasets used in this chapter's experiments are described in section 5.3. A full breakdown of all the data used in this investigation can be viewed in table 8. The non-fascist samples are comprised of a random selection of documents acquired from the non-fascist, non-hateful *Reddit* dataset (see section 5.2). The quantity of non-fascist samples was selected in a manner as not to induce imbalanced classes.

Dataset	**Validation technique**	**Fascist-samples**	**Non-fascist samples**
Gold	Cross-validation (5-fold)	500 (train and test)	664 (train and test)
Shuffled	Hold-out (70:30)	1117 (train) 151 (test)	1050 (train) 449 (test)
SR	Hold-out (70:30)	1446 (train) 137(test)	1422 (train) 629 (test)
Synthetic	Hold-out (70:30)	17396 (train) 141 (test)	17396 (train) 141 (test)

Table 8: Datasets breakdown binary classification.

7.2 Results

Due to the volume of results generated in this investigation, it was decided that the tables displaying the classification reports for each model and dataset should be located in the reports appendices. The location of each report is as follows:

Gold datasets results: page 61, table 13.
Shuffled dataset results: page 61, table 14.
SR dataset results: page 62, table 15.
Synthetic dataset results: page 62, table 16.

7.3 Discussion

In general, most models were easily able to distinguish between the samples belonging to the fascist and non-fascist classes and out-performed expectations. All models achieved a higher or equal F1-score for their classification of the non-fascist text than opposed to fascist text, however, the contrast was not always extensive. This came as somewhat of a surprise, as from our findings in previous hate speech studies, it was expected that most models would have experienced a more difficult task in classifying the fascist documents than found. Nevertheless, this is not to say that all trials were a success, with certain models underperforming at the given task. This is illustrated in table 9.

		Worst Performing Models: Fascist vs Non-Fascist Speech									
		Fascist			Non-fascist			Weighted-Average			Acc.
		P	R	F1	P	R	F1	P	R	F1	A
Random Forest	Tf-idf word n-grams with Synthetic dataset	1.00	0.10	**0.18**	0.53	1.00	0.69	0.76	0.55	0.44	0.55
	Tf-idf char n-grams with Synthetic dataset	1.00	0.28	**0.43**	0.58	1.00	0.73	0.79	0.64	0.58	0.64
Logistic Regression	Word embeddings with SR dataset	0.57	0.85	**0.68**	0.96	0.86	0.91	0.89	0.86	0.87	0.85
	Paragraph Vector with SR dataset	0.56	0.74	**0.64**	0.94	0.87	0.90	0.87	0.85	0.86	0.86

Table 9: Worst performing models per feature extraction method for fascist document classification.

One possible explanation for the high results may originate from the test data that was used in the experiments. Although we had gone to great lengths to generate higher quantities of training data for our models to learn on, it was not possible to create new augmented or synthetic test data, as this would have not been reflective of real-world data a model would be expected to predict on. Consequently, each model only had roughly 150 test samples belonging to the fascist class to be evaluated with. Therefore, if these limited number of test samples too closely resembled the original training data, it may explain why some models performed to the extent they did.

Nevertheless, this critic only applies to three of our four datasets. The *Gold* dataset was evaluated by means of a 5-fold cross-validation, in which all of its data was used for both training and test purposes. As the alternative datasets were derived from this dataset, then perhaps a more likely explanation could be a unique contrast in the language that was used in either class.

7.3.1 Algorithm Evaluation

When assessing which of the machine learning algorithms was the best performing, the Linear-SVC was the winner overall (by F1-score). The mean F1-scores for strictly the classification of fascist documents, computed for each of the algorithms across all four datasets for every feature combination ranked from best to worst are as follows; Linear-SVC (0.83%), Logistic Regression (0.77%), Random Forest (0.72%). This along with the mean precision and recall scores is further illustrated below in figure 15.

In terms of overall performance for a distinct feature combination, all of the top performing models were achieved through the Linear-SVC. We selected the Support-vector Machine algorithm due to its known ability to work effectively with small to medium sized datasets (Gèron, 2019), and this was

certainly reflected in the results. It performed exceedingly well with the *Gold*, *Shuffled* and *SR* datasets, particularly in combination with the TF-IDF word and character n-gram features. What was more striking perhaps, was that the Linear-SVC attained the highest results for the larger *Synthetic* dataset as well, where it achieved a weighted average F1-score of 0.91% when aggregated with both word embeddings and Paragraph vector features (see table 10 in 7.4).

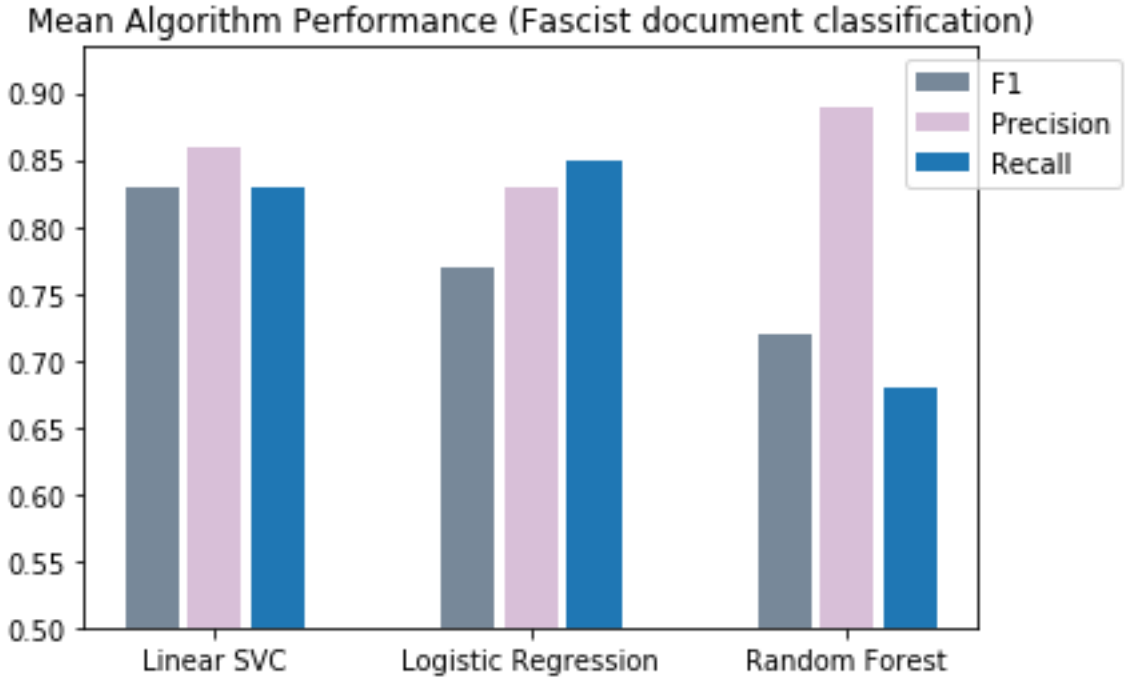

Figure 15: Mean algorithm performances respecting the binary classification of fascist text documents across all datasets and features.

Conversely, one particular surprise was the performance of the Random Forest implementations. We had selected the Random Forest model for its reputation of handling large sized datasets (James et al., 2017), unfortunately however, this proved not to be the case in this investigation. It achieved consistent scores with the small to mid-range sized datasets, yet, it failed to scale to the demands of the larger *Synthetic* dataset, where it proved by far the worst performer. This was due to the Random Forests manner of having extremely high precision yet terrible recall scores for fascist text classification, and vice-versa for non-fascist text. For some reason the *Synthetic* dataset seemed to amplify this shortcoming, as visible by its F1-scores of 0.18% and 0.43% for the classification of fascist text when combined with TF-IDF word and character n-gram features respectively (see table 9). A possible explanation for this may be the artificial nature of the *Synthetic* dataset's samples, or even, the inability to tune the models using the synthetic data's parameters (see 6.5).

Logistic Regression achieved consistently good results, proving to be very capable across the range of feature and dataset combinations. This was only masked, however, by the slight superiority in the implementation of the Support-vector Machine.

7.3.2 Feature and Dataset Evaluation

The mean F1-scores for the classification of fascist documents based on their features, ranked from best to worst, was calculated as; TF-IDF character n-grams (0.83%), word embeddings (0.81%), TF-IDF word n-grams (0.80%), Paragraph vector (0.75%). This is further illustrated in figure 16 along

with the features precision and recall averages. Overall, there was no unequivocal stand-out among the feature selection methods, yet both the word embedding and TF-IDF character n-gram features outperformed the rest, with the latter just edging it for overall performance.

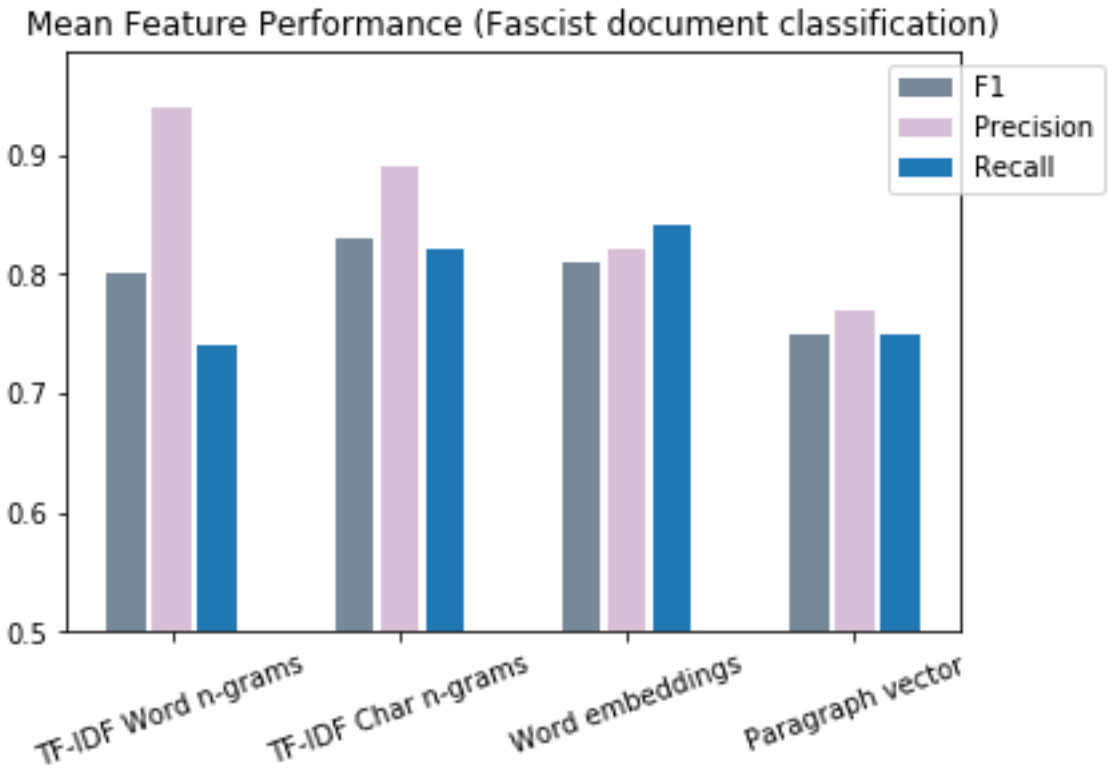

Figure 16: Mean feature performances respecting the binary classification of fascist text documents across all datasets and algorithms.

Nevertheless, as would be expected certain features excelled in the environment of specific datasets and amongst varying quantities of data. Notably, the *Synthetic* dataset outperforms all others when used in parallel with Paragraph vector and word embeddings. A plausible explanation for this can be derived from the knowledge that both implementations utilize a neural network to learn their word associations. This is of particular relevance for the Paragraph vector models. Unlike our adopted word embedding approach that exploits pretrained word vectors, the Paragraph vector models train their word associations exclusively on the vocabulary provided in the given datasets. Notoriously, neural networks require vast amounts of data to learn tasks from. Therefore, the *Synthetic* datasets ability to provide thousands of extra training samples explains its superior display when combined with the Paragraph vector and word embedding features. Notably however, Paragraph vector was on average the poorest performing of all features. A possible explanation for this may arise from the size of the documents it was tasked with classifying which were mostly relatively small in length, and as such, they may have been challenging to form useful or adequate paragraph vectors from.

The success of the TF-IDF implementations provide further evidence as to why it is regarded as one of the most successful feature extraction techniques concerning NLP. Both word and character n-gram models had the highest precision rates out of all features. The TF-IDF word n-gram features, which applied unigram, bi-grams and tri-grams had the poorest recall on average however, resulting in its overall impeded F1-score (see figure 16). This poor recall score only seems to have arisen when

used with the *Synthetic* dataset however, and therefore this was only an isolated occurrence which affected its overall average.

Overall, the TF-IDF features were best received with the *Gold* and *Shuffled* datasets. The highest F1-score for fascist document classification for a TF-IDF character n-gram model was seen in combination with the Linear-SVC and the *Gold* dataset, scoring 0.95%. Alternatively, the best F1-score for a TF-IDF word n-gram model was similarly observed in partnership with the Linear-SVC, this time with the *Shuffled* dataset. This particular model had the highest weighted-average F1 score of all of our models, at 0.97% for both classes. An explanation for TF-IDFs heightened performance with the *Shuffled* dataset may stem from the expanded repetitions of similar words and terms this dataset was able to provide. As TF-IDF partially represents features in accordance with their term frequency occurrence, this would have most likely aided its process.

The *SR* (Synonym Replacement) dataset, in general, was the worst performing of the small to medium sized datasets. It performed reasonably well with the TF-IDF features, however, when applied with word embedding and Paragraph vector features, precision scores suffered greatly. The only anomaly from this trend in the *SR* dataset was seen with word embedding features combined with a Random Forest, which resulted in a respectable F1-score of 0.80% for fascist document classification. A possible reason for the subpar performance may be that the synonyms generated at random were not as representative as their precursors as we may have hoped for. For this reason, perhaps it would have been more fitting to have employed a predefined threshold that would have only selected synonyms that were considered to align more closely to the original word.

7.4 Model Selection and Summary

In this chapter, we implemented several machine learning algorithms and feature extraction techniques on a range of datasets and evaluated their suitability to classify binary, fascist vs. non-fascist text. In doing so, we demonstrated that machine learning techniques can adequately distinguish fascist from regular speech. Several models proved more than capable to accomplish this task, with table 10 displaying the best models for each of the different feature extraction techniques. We now take these four models forward to a multiclass classification investigation in the subsequent chapter. In which, we will incorporate hate-speech as an additional category.

		Best Performing Models: Fascist vs Non-Fascist Speech									
		Fascist			Non-fascist			Weighted-Average			Acc.
		P	R	F1	P	R	F1	P	R	F1	A
Linea-SVC	Tf-idf word n-grams with Shuffled dataset	0.97	0.93	**0.95**	0.97	0.99	0.98	0.97	0.97	0.97	0.97
	Tf-idf char n-grams with Gold dataset	0.94	0.95	**0.95**	0.96	0.96	0.96	0.95	0.95	0.95	0.95
	Word embeddings with Synthetic dataset	0.93	0.89	**0.91**	0.89	0.93	0.91	0.91	0.91	0.91	0.91
	Paragraph Vector with Synthetic dataset	0.96	0.85	**0.90**	0.86	0.97	0.91	0.91	0.91	0.91	0.91

Table 10: Best performing models per feature extraction method for fascist document classification.

CHAPTER 8: Multiclass Classification (Fascist vs. hate-speech)

Introduction

Having demonstrated that we can successfully use machine learning techniques to classify binary fascist vs. non-fascist speech, we now take our best performing models from the previous chapter forward to a multiclass classification investigation. In this analysis, hate will be incorporated as an additional third class representing hate speech. The rationale for conducting this experiment is twofold. Foremost, we wish to see if it is possible to build a machine learning model that can accurately classify fascist vs. hate speech. Yet moreover, through investigating this possibility, we also seek to discover whether we can use these machine learning models to demonstrate a clear distinction between the two styles of language. To conduct these experiments, we apply the exact same methodology as detailed prior in chapter 6.

8.1 Data Description

To facilitate this investigation, we required a suitable number of documents containing examples of hate speech. Fortunately, finding such data is far less convoluted than the efforts made to source the fascist speech. A publicly available dataset from a hate speech study conducted by Qian et al. (2019)[3] that sought to encourage techniques in countering online hate speech was identified to use for the hate class. It contains messages from Reddit posts that had been flagged by moderators as containing hateful language. As such, the data fittingly complements the style of our non-fascist documents which are also sourced from Reddit message posts.

For means of clarity, we change the label of the previous 'non-fascist' class to 'neither' in this investigation. This is to represent language that is classed as neither fascist or hate speech. We then concatenate the hate samples as an additional third class, labelled hate, onto our original *Gold*, *Shuffled* and *Synthetic* datasets that were selected for use in this investigation. In accordance with the fascist documents, the hate samples are also augmented via sentence reordering in the *Shuffled* dataset, and further, have synthetic instances generated in the *Synthetic* dataset. A full breakdown of the data used in this investigation is highlighted in table 11.

Dataset	**Validation technique**	**Neither**	**Fascist**	**Hate**
Gold	Cross-validation (5-fold)	664 (train and test)	500 (train and test)	562 (train and test)
Shuffled	Hold-out (70:30)	1050 (train) 449 (test)	1117 (train) 151 (test)	1132 (train) 169 (test)
Synthetic	Hold-out (70:30)	17396 (train) 141 (test)	17396 (train) 141 (test)	17396 (train) 141 (test)

Table 11: Datasets breakdown multiclass classification.

[3] https://github.com/jing-qian/A-Benchmark-Dataset-for-Learning-to-Intervene-in-Online-Hate-Speech

8.2 Data Analysis

Prior to conducting this investigation, we decide to further explore the data that is contained within the fascist and hate classes through the creation of visual representations. The intention is to convey the commonly used words and phrases that are specific to each class. Through this means, we hope to observe contrasts or parallels that may be present in the two styles of language. We discuss the content of this sections data analysis in 8.4.

Figure 17: Word clouds featuring most frequently occurring words in the fascist samples (left) and the hate samples (right).

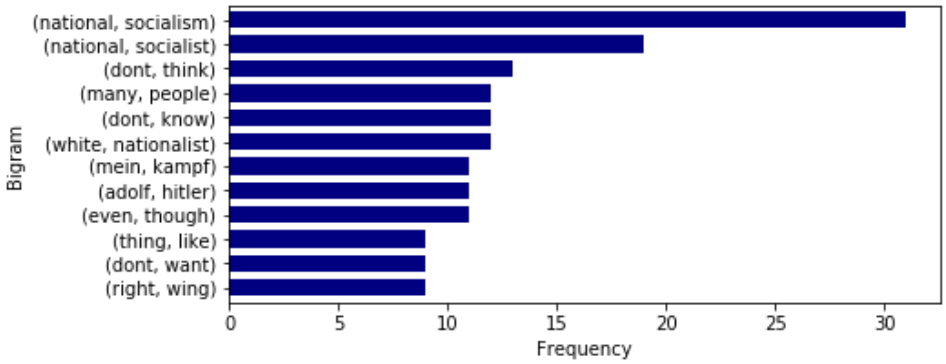

Figure 18: The most frequently occurring bi-grams in the **fascist** documents.

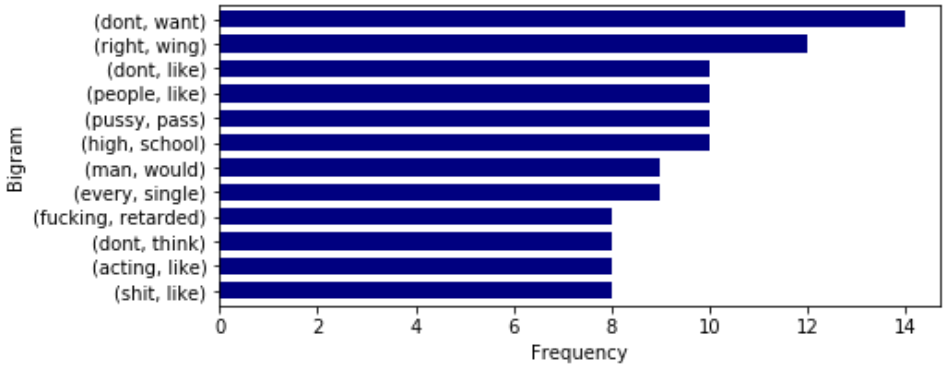

Figure 19: The most frequently occurring bi-grams in the **hate** documents.

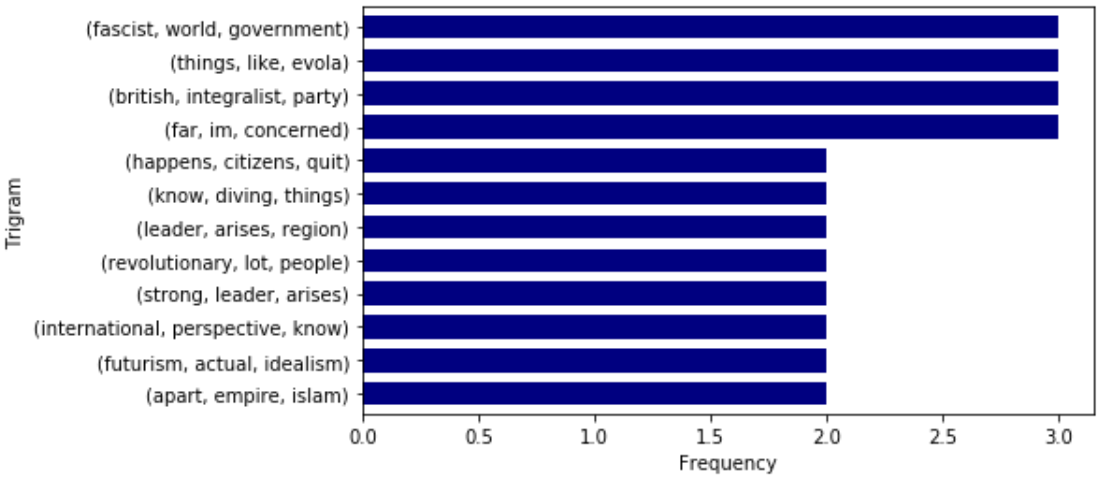

Figure 20: The most frequently occurring tri-grams in the **fascist** documents.

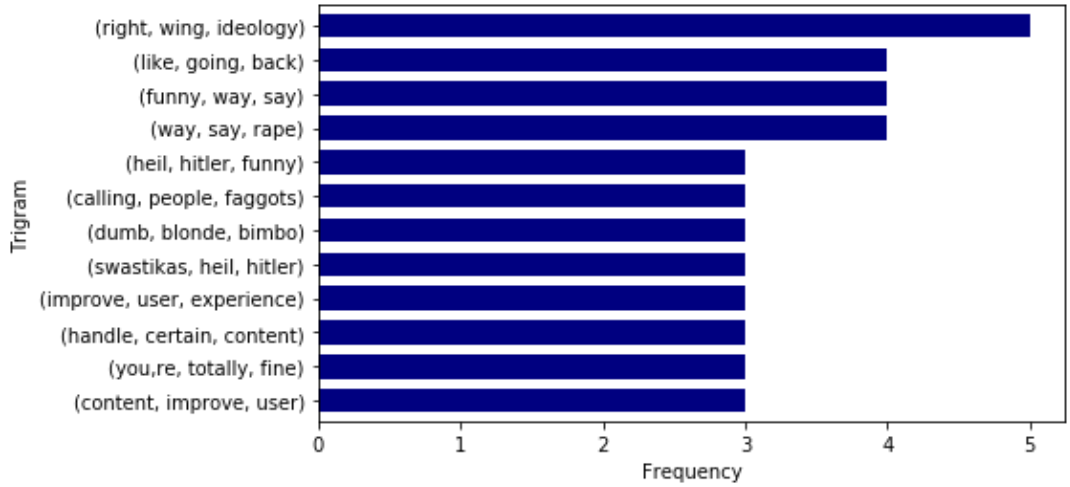

Figure 21: The most frequently occurring tri-grams in the **hate** documents.

8.3 Results

In this investigation, the four models displayed in table 10 (see 7.4) were selected to perform the multiclass classification. We use confusion matrices and a classification report displayed below to convey the results.

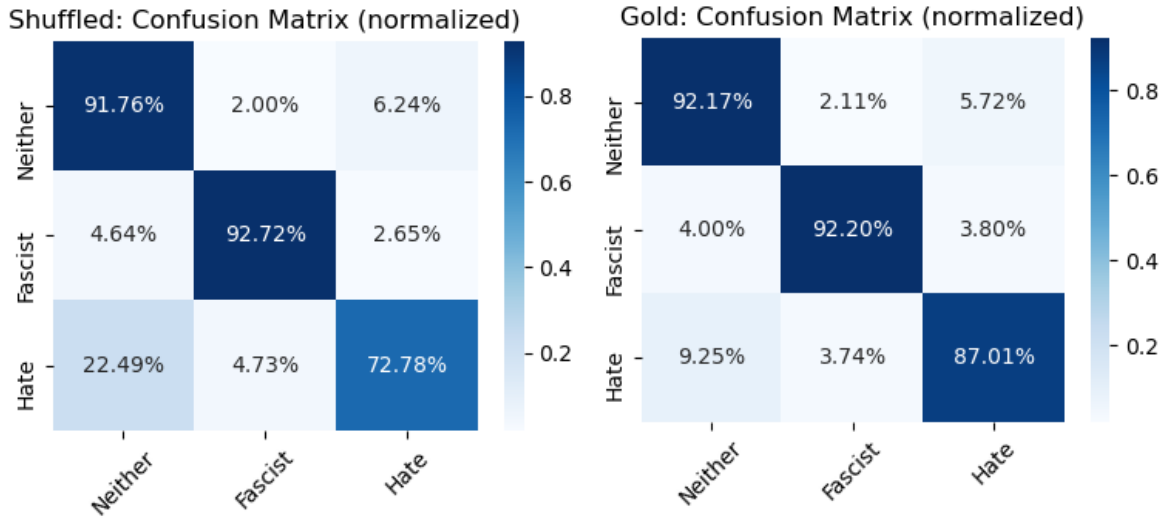

Figure 22: Confusion matrices; TF-IDF Word n-grams and Linear-SVC with Shuffled dataset (left). TF-IDF character n-grams and Linear-SVC with Gold dataset (right).

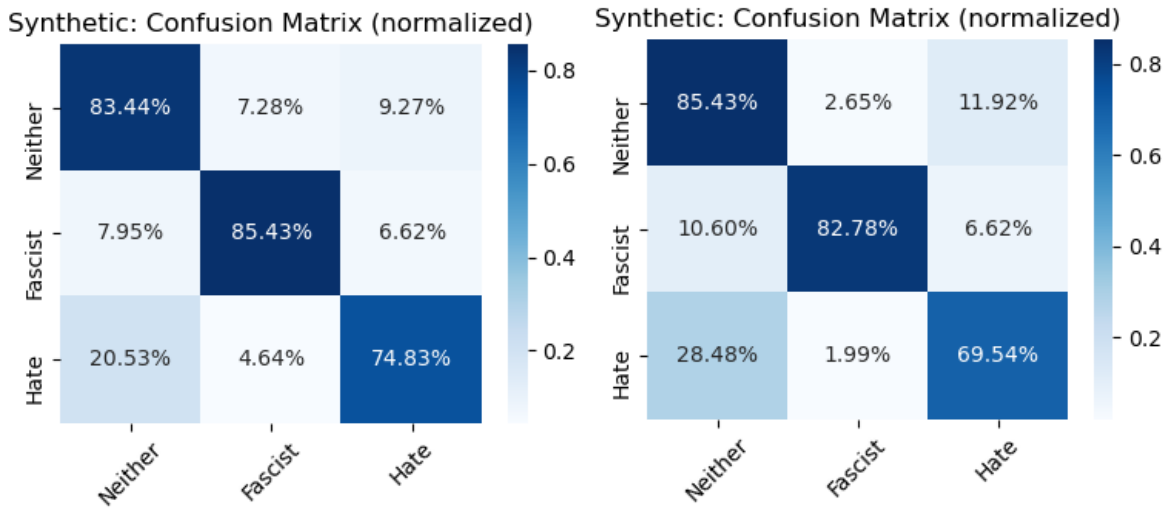

Figure 23: Confusion matrices: Word embeddings and Linear-SVC with Synthetic dataset (left). Paragraph vector and Linear-SVC with Synthetic dataset (right).

		Classification Report: Neither vs. Fascist vs. Hate									
		Neither			Fascist			Hate			Acc.
		P	R	F1	P	R	F1	P	R	F1	A
Linear-SVC	Tf-idf word n-grams with Shuffled dataset	0.90	0.92	0.91	0.89	0.93	0.91	0.79	0.73	0.76	0.88
	Tf-idf char n-grams with Gold dataset	0.89	0.92	0.91	0.93	0.92	0.93	0.90	0.87	0.88	0.90
	Word embeddings with Synthetic dataset	0.75	0.83	0.79	0.88	0.85	0.87	0.82	0.75	0.78	0.81
	Paragraph Vector with Synthetic dataset	0.69	0.85	0.76	0.95	0.83	0.88	0.79	0.70	0.74	0.79

Table 12: Multiclass classification report: Fascist vs. hate speech.

8.4 Discussion

In chapter 4 section 4.3, 'Distinctions vs. Hate speech', we hypothesised that several of our defining attributes of fascist speech may likely intersect with certain areas of hate speech. For example, one would assume that attributes such as 'racism', 'xenophobia' and 'anti-feminism' that we identify as fascist speech, are almost certainly intertwined with hateful language. On the other hand, there are particular attributes that we would associate as being more uniquely specific to fascism. Instances of which would be language pertaining to attributes such as 'totalitarianism', 'imperialism' and 'ultra-nationalism'. Consequently, we anticipated that the extension of the additional hate class would induce conflict into the models when attempting to categorize documents that possess these overlapping attributes.

On the contrary however, this proves not to be the case. The confusion matrices (figures 22 and 23) visible in section 8.3 illustrate that the models are still highly capable of distinguishing between the two styles of language, especially the models with TF-IDF features. This is further demonstrated in the classification report in table 12. Each model struggled with the classification of hate speech above the rest, whereas fascist speech was more comfortably identified. The best performing model was the Linear-SVC combined with the TF-IDF character n-grams features with the *Gold* dataset. This model achieved an overall mean accuracy result of 0.90%, and F1-scores of 0.93% and 0.88% for the classification of fascist and hate documents respectively. The models utilizing feature embeddings, namely word embeddings and Paragraph vector features, performed moderately worse than their TF-IDF competitors. In total there was roughly a swing in 0.10% accuracy between these models. This may be due the features themselves, or rather, the synthetic nature of data that these models were combined with, which may not have scaled so well with an additional class to consider.

From the results of this investigation we derive the inference that a clear distinction between fascist and hate speech exists. This could stem from a contrast in terminology that is specific to each style of language. Support for this argument is evident in the divergent Word Clouds generated for each class, which can be viewed in section 8.2 (figure 17). This illustrates a clear difference in the commonly used vocabulary which is unique to each class.

Nevertheless, we must highlight one deviation concerning the data analysis from section 8.2 that goes against this assertion. In the most frequently occurring tri-grams for the hate class (see figure 21), we see combinations of; (right, wing, ideology), (heil, hitler, funny) and (swastikas, heil, hitler).

On first observation one would most likely assume this terminology to be associated with fascist principles, and yet, these word grams were derived from the hate class. This would indicate that hate speech, to a certain degree, is still fuelled by extremist ideologies. However, in one tri-gram, the inclusion of the word ‘funny’ following ‘heil’ and ‘hitler’, likely indicates that the context in which these terms are used is markedly different from fascist speech. As if we contrast that with the tri-gram, (strong, leader, arises) from the fascist data (see figure 20), we can construe a likely deviation within the underlying sentiment of these word phrases.

Moreover, another reason for the distinction between the two languages may arise from a lack of hate speech samples in our dataset that correlate to any of our defining fascist attributes. We decide to assess this possibility further. To do so, we inspect each of the documents belonging to the hate class and record any that include language associated with our defining fascist attributes (see table 6) as ‘both’ (both fascist and hate). For this argument to hold weight, we would expect a lack of these examples to be recorded. Out of the 562 documents in the hate speech class, we identify 97 of which to have notable language corresponding to fascist attributes. These were namely the attributes of; Anti-feminism, Anti-Semitism, Glorification of violence, Nativism, Racism, Supremacism and Xenophobia. We calculate this to comprise roughly 17% of all of the hate samples.

It is not clear whether given a larger sample size of hate-speech documents how much this percentage would deviate. Nonetheless, we would still expect a larger misclassification rate between the two classes given this existence. Therefore, an explanation for the absence of this may arise from a confliction in writing styles, and as stated previously, a divergence in both the phraseology and terminology the documents are composed in. To inspect this possibility, we evaluate documents from the hate speech classes that we identified as ‘both’ and compare them to fascists documents labelled with the same attribute. We conceal sensitive words with the asterisk character.

Below is an example of a document in the **hate** class we identify as having racist and supremacist language…

*“In an ironic twist, dumb f**king hillbilly white people like yourself have become the country's lowest common denominator. The most amusing part is you're too stupid to even hide it anymore.”*

Conversely, the following is a document in the **fascist** class that we also establish as containing racist and supremacist language…

*“I don't know if you've seen many of my posts on this forum yet, but I do believe in the inferiority of the negr**d race and the overall superiority of the caucasoid race, and in particular the ethnic British people out of all white ethnic groups.”*

We also examine a document in the **hate** class containing anti-Semitist remarks…

“*I thought the same thing until I recognized the roots of most of our problems. They're jewish, you dig to the bottom of anything rotten or evil and you find a jew at the root. It's scary to be on this side, you're alone with a bunch of other truth seeking goys who've been preemptively f**ked up by, you guessed it, the jews.”*

Below we see a document in the **fascist** class with language assigned as anti-Semitism…

"*You know the Jews even claim that the Legionaries made pogroms against them which never actually happened. In truth, it is the Jews who have been persecuting my people along with the Germans for centuries.*"

The following is a document from the **hate** class with anti-feminist remarks…

"*Having a manipulative, scheming woman as a secondary antagonist didn't come across as excessively woke either. It felt like a touch, though it felt more like how sjw feminist c**ts actually act rather than how they THINK they act. Only with more skill."*

Finally, we observe a document from the **fascist** class with anti-feminist language…

"I did believe that was the case; feminism has always been a fringe group, but the arguments catch on a lot more frequently today while ironically women still take up their natural roles; there's no glass ceiling, it's simply because women don't want those higher positions, work less hours or are not qualified for them."

One evident and recurring observation regarding documents from the hate class was the occurrence of curse words and derogative slurs within the documents. This was considerably less apparent in the fascist documents. A reason for this may be that many of the hate documents were extracted from message posts in which people with differing opinions were conflicting with one another over certain affairs. Whereas the majority of the fascist documents were composed by like-minded individuals who largely agreed on the subject matters being conversed. In addition, although it may not be so apparent in the above examples, a much larger quantity of documents belonging to the hate class contained grammatical errors as opposed to the fascist class. Both of these characteristics may have aided a model's task in distinguishing between the two categories of language.

8.5 Misclassification Analysis

An interesting observation provided by the confusion matrices (see 8.3) was the misclassification distribution of the classes. It would be rationale to anticipate that the majority of fascist speech would be misclassified as hate speech, and vice-versa. Despite this, we actually observe the occurrence of the complete opposite, as in each of our four models, the majority of the fascist and hate documents were misclassified as 'neither'. Conclusions to draw from this can be difficult to establish. In this subchapter we examine misclassified documents for each class in the attempt to gain some further insight as to why this may have transpired. As the model with TF-IDF character n-gram features was our most successful implementation, the misclassified documents listed in this subchapter will be those that this model failed to classify.

8.5.1 Neither Misclassification

The following examples are of documents belonging to the neither category that were wrongly misclassified.

The subsequent document was predicted as **fascist**:

"*Yeah, because 6 million Jewish people just trolled everyone during WW2 by making everyone think they were killed in gruesome ways by a madman fascist and his group of murderous thugs.*"

This comment appears to have been made in a sarcastic context, a very difficult sentiment for a machine learning model to pick up on. Its reference to fascism likely led to its misclassification.

Below we see a document that was precited as **hate**:

*"I'm a gay man. I don't find women making out to be gross at all. People honestly need to get the f**k over human sexuality."*

The inclusion of the curse word may have resulted in this misclassification. Alternatively, Dixon et al. (2018) found that the high use of identity terms such as 'gay' in their toxic comments led to their classifier overgeneralizing on them in non-toxic comments.

8.5.2 Fascist Misclassification

Next we examine documents of the fascist class that were incorrectly misclassified.

The successive document was misclassified as **neither**:

"Sieg Heil,
you mentioned in the Esoteric Hitlerism Thread a research group. Can you give me some information how I can get involved in this?"

Potentially there were too few occurrences of the words relating to Nazism in other fascist documents leading to this misclassification, specifically 'Sieg Heil'.

Next we observe a document that was predicted as **hate**:

*"Exactly, he is a f**king re**rd and anything is not capitalist is automatically COMMUNIST and Mussolini was a capitalist sweetheart who did his best to deffend the free market but teh evil syndicalists held him back."*

We would assume that the offensive language in addition to the grammatical errors resulted in this particular misclassification for the reasons stated in 8.4.

8.5.3 Hate Misclassification

Finally, we analyse the misclassification of hate documents.

The following is an example of a document that was predicted as **neither**:

*"All men have d*cks, therefore, all men must be d*cks."*

This is an instance of sexist language, yet, the classifier failed to recognize it. Again, this could be due to a lack of documents with similar terminology in the relevant class that our model had to train on.

Below we examine a document that was assigned to the **fascist** class:

*"You're re**rded. The other poster was logical, unlike what you spew. Europe is a great place to live in, but you're wholly ignorant - willfully, I shall add - to think European imperialism was anything but the decadence of the human timeline. Of course, you think you have to believe in imperialism in order to be a nationalist. Brainlet."*

Certain key terms relating to fascist speech such as 'imperialism' and 'nationalist' may have resulted in this particular misclassification.

8.6 Tweet analysis

A useful way to determine how well a model can generalize to data outside of its corpus is to test it on real-world instances. In recent weeks, British musician Wiley made headlines for the wrong reasons after posting a flurry of anti-Semitic tweets (posts on Twitter). Subsequent protests and a public outcry resulted in Twitter suspending his account and led to a police investigation (BBC, 2020).

For the academic purposes of trialling this studies machine learning techniques, we have our best performing binary classifier (see 7.6) and multiclass classifier (see 8.3) predict on several of the identified tweets. Following the suspension of the users account on the grounds of anti-Semitic content, we would hope that the classifier predicts the fascist category, or at least the hate category for the multiclass model. This shall hopefully provide an insight as to how well our models can adapt to previously unseen data from a different social networking source.

Tweet 1: "*Jewish people you make me sick and I will not budge hold this corn.*"

Binary classifier prediction: 'Non-fascist'
Multiclass classifier prediction: 'Neither'
Analysis: In this instance both classifiers predict this as a non-offensive document. Upon conducting further research regarding the tweets, it was later found that the phrase 'hold this corn' is actually a British slang term with the connotation, to 'be shot' or to 'receive bullets' (The Scotsman, 2020). This highlights a very troublesome obstacle that social network companies face when deploying automated classification detectors for such tasks. To expect our own machine learning system to recognize such an atypical term would be ambitious, especially given our stated data difficulties. For this reason, our classifiers failure in this scenario is understandable. The only way to overcome this failure is to accumulate more data with the religious target group in the presence of the aggressive slang term.

Tweet 2: *"JEWISH PEOPLE ARE RACIST THEY HIDE."*

Binary classifier prediction: 'Fascist'
Multiclass classifier prediction: 'Fascist'
Analysis: On this occasion both classifiers make the correct call. We would expect so in this circumstance, as the tweet specifies a clear target group 'Jewish people' and a clear insult against them by generalizing them as a 'racist' community.

Tweet 3: "*Infact there are 2 sets of people who nobody really wanted to challenge #Jewish & #KKK but being in business for 20 years you start to understand why*"

Binary classifier prediction: 'Non-fascist'
Multiclass classifier prediction: 'Neither'

Analysis: Again, our classifiers incorrectly assign this document to the non-offensive class, the reasoning as to why this likely is may be more difficult to establish. The context is anti-Semitic, as it is making a comparison between the Jewish community and the overtly racist Ku Klux Klan organization. As such, it is a demeaning statement intended to cause offense to the target group. If we examine the syntactic structure and terminology used within the document however, we can understand why this would be difficult for a machine learning system to interpret. There are no 'trigger' words (i.e. offensive slurs or evidently fascist language) contained in the document. Therefore, in order to have predicted correctly, our classifiers would first be expected to recognize that a comparison was being made, and more, that the comparison was between a hate group and a target group for the purpose of causing insult. An incredibly difficult task.

8.7 Summary

In this chapter, we sought to establish if we could train machine learning models to accurately perform multiclass classification concerning fascist vs. hate speech. We conducted the investigation with several different models, and in general, all of which achieved notable results (see 8.3). All models found hate speech as the most difficult class to correctly predict, with the majority of instances being misclassified to the neither target category. The best performing model achieved a mean accuracy score of 0.90% and was composed of TF-IDF character n-gram features with a Linear-SVC utilizing the *Gold* dataset. In the discussion of the results (see 8.4), we determined that we had demonstrated a distinction between fascist and hate speech, in which we used evidence from the data analysis (see 8.2) to further help demonstrate this. During the Tweet analysis (see 8.6), our binary and multiclass classifiers struggled to generalize to unseen data from a new networking source. Through this process, we shed light on some of the difficulties that machine learning models face when deployed with real world data in the field of NLP.

CHAPTER 9: Conclusion

The overarching aim of this study was to determine if machine learning techniques could be used to detect fascist text in social networks. In order to achieve this objective, we first formulated a set of defining attributes that we determined to epitomize contemporary fascist speech. Based on these defining attributes, we created a number of diverse datasets that could be used to train classifiers. We then conducted several investigations to realize the projects goals.

9.1 Achievements and Contribution

We began by supervising a binary fascist vs. non-fascist classification investigation. The primary purpose of this experiment was to first establish that the classification of fascist text was indeed possible, as to the best of our knowledge, no such research regarding fascist speech has been conducted before. Overall, this proved to be a success as we created several successful models capable of achieving the task at hand. The most successful of these models, was the TF-IDF word n-gram features combined with the Linear-SVC evaluated with the *Shuffled* dataset. This model achieved a 0.95% F1-score for the classification of fascist documents and 0.97% weighted average F1-score for both classes.

During this this initial investigation, we also intended to implement and evaluate a range of feature extraction techniques and machine learning algorithms in their ability to classify fascist text. We found the Support Vector Machine implemented in the form of a Linear-SVC to be the most optimal algorithm in terms of overall mean F1-score. Conversely the Random Forest was the least successful of the algorithms, and was especially poor with the *Synthetic* dataset in which its recall greatly suffered. It was also confirmed that the word embedding and Paragraph vector features were most successful in partnership with the larger *Synthetic* dataset, which provided it thousands of data samples to establish suitable word associations from. The TF-IDF features on the other hand, were more suited to the smaller to medium sized datasets.

This research also attempted to answer the question as to whether or not we could use machine learning techniques to establish if a clear distinction between fascist and hate speech exits. To fulfil this question, we performed multiclass classification in which we incorporated hate speech as an additional class. The results of this experiment illustrated that the intersect between the two classes of speech was not as apparent as predicted, and our models were fairly capable at distinguishing between the two types of language. Our best multiclass classifier proved to be the TF-IDF character n-gram features with a Linear-SVC in conjunction with the *Gold* dataset. This model attained F1-scores of 0.93% and 0.88% for the classification of fascist and hate documents respectively, with an overall mean accuracy result of 0.90%.

9.2 Limitations and Future Work

It is hoped that this research has provided an initial building block in an inquest to determine the ability of machine learning techniques to classify fascist text. One hindrance faced was the lack of contemporary labelled data containing fascist speech. The leaked message posts from the *Iron March* dataset allowed us to construct our own Gold standard dataset, from which we used augmentation techniques to expand upon. However, only 500 original fascist samples could be derived from this source. In contrast to other hate speech studies that have had the opportunity to use thousands of original data samples for their classes, our fascist data seems somewhat lacking in quantity if not

quality. This limitation was exemplified in section 8.6, in which our models struggled to generalize when deployed on real world data from a different social networking source.

Concerning future work, an interest that stemmed as a result of conducting this research was the idea of analysing the impact that fascist regimes had on their respective nations. For example, would Italian citizens be more likely to use fascist speech in their social networking conversations then say native French speakers? By doing so, we may be able to answer the question as to how impactful or successful fascist regimes were at indoctrinating their youth (and if this influence has passed on to future generations). An apparent difficulty of this would be quantifying what constitutes fascist speech for respective nationalities, in addition to various censorship laws imposed in certain regions.

References

Almashraee, M., Monett, D. and Paschke, A., (2014). The Gold Standard in Corpus Annotation. In: *5th IEEE Germany Student Conference*. [online] Available at: <https://pdfs.semanticscholar.org/124f/69e9549535f07bc77f29ff91b8330aa429a0.pdf> [Accessed 18 July 2020].

Badjatiya, P., Gupta, S., Gupta, M. and Varma, V., (2017). Deep Learning for Hate Speech Detection in Tweets. *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion*, [online] Available at: <https://dl.acm.org/doi/10.1145/3041021.3054223> [Accessed 5 June 2020].

BBC News. (2019). *Daily Stormer Founder 'Should Pay $14M' In Damages, Judge Says*. [online] Available at: <https://www.bbc.co.uk/news/technology-49003766> [Accessed 17 July 2020].

BBC News. (2020). *Wiley: Anti-Semitism Row Prompts 48-Hour Twitter Boycott*. [online] Available at: <https://www.bbc.co.uk/news/technology-53553573> [Accessed 18 August 2020].

Bravo-Marquez, F., Frank, E. and Pfahringer, B., (2016). Building a Twitter opinion lexicon from automatically-annotated tweets. *Knowledge-Based Systems*, [online] 108, pp.65-78. Available at: <https://doi.org/10.1016/j.knosys.2016.05.018> [Accessed 6 July 2020].

Brown, P., De Souza, P., Mercer, R., Della Pietra, V. and Lai, J., (1992). Class-based n-gram models of natural language. In: *Computational Linguistics, Volume 18(4)*. [online] pp.467–480. Available at: <https://www.aclweb.org/anthology/J92-4003> [Accessed 28 July 2020].

Burnap, P. and Williams, M. (2015). Cyber Hate Speech on Twitter: An Application of Machine Classification and Statistical Modelling for Policy and Decision Making. *Policy & Internet*, [online] Volume 7(2). Available at: <https://doi.org/10.1002/poi3.85> [Accessed 1 July 2020].

Chawla, N., Bowyer, K., Hall, L. and Kegelmeyer, W., (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, [online] 16, pp.321-357. Available at: <https://arxiv.org/abs/1106.1813> [Accessed 30 June 2020].

Clement, J., (2020). *Daily Social Media Usage Worldwide*. [online] Statista. Available at: <https://www.statista.com/statistics/433871/daily-social-media-usage-worldwide/> [Accessed 27 June 2020].

code.google.com. (2013). *Word2vec (Pre-Trained Word and Phrase Vectors)*. [online] Available at: <https://code.google.com/archive/p/word2vec/> [Accessed 7 July 2020].

Daniels, J., (2009). *Cyber Racism: White Supremacy Online and the New Attack on Civil Rights*. Plymouth: Rowman & Littlefield Publishers Inc.

Davidson, T., Warmsley, D., Macy, M. and Weber, I., (2017). Automated Hate Speech Detection and the Problem of Offensive Language. In: *Proceedings of the Eleventh International AAAI Conference on Web and Social Media*. [online] Available at: <https://arxiv.org/pdf/1703.04009.pdf> [Accessed 7 July 2020].

Del Vigna, F., Cimino, A., Dell'Orletta, F., Petrocchi, M. and Tesconi, M., (2017). Hate me, hate me not: Hate speech detection on Facebook. In: *Italian Conference on Cybersecurity (ITASEC17)*. [online] Available at: <http://ceur-ws.org/Vol-1816/paper-09.pdf> [Accessed 1 July 2020].

Dixon, L., Li, J., Sorensen, J., Thain, N. and Vasserman, L., (2018). Measuring and Mitigating Unintended Bias in Text Classification. *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, [online] Available at: <https://doi.org/10.1145/3278721.3278729> [Accessed 8 July 2020].

Djuric, N., Zhou, J., Morris, R., Grbovic, M., Radosavljevic, V. and Bhamidipati, N., (2015). Hate Speech Detection with Comment Embeddings. *Proceedings of the 24th International Conference on World Wide Web - WWW '15 Companion*, [online] Available at: <https://doi.org/10.1145/2740908.2742760> [Accessed 11 July 2020].

Domingos, P., (2012). A few useful things to know about machine learning. *Communications of the ACM*, [online] 55(10), pp.78-87. Available at: <https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf> [Accessed 23 August 2020].

Duarte, N., Llanso, E. and Loup, A., (2018). Mixed Messages? The Limits of Automated Social Media Content Analysis. In: *Conference on Fairness, Accountability, and Transparency*. [online] PMLR. Available at: <http://proceedings.mlr.press/v81/duarte18a.html> [Accessed 14 July 2020].

Esuli, A. and Sebastiani, F., (2006). SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining. In: *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. [online] Genoa, Italy: European Language Resources Association (ELRA), pp.417-422. Available at: <http://www.lrec-conf.org/proceedings/lrec2006/pdf/384_pdf.pdf> [Accessed 10 July 2020].

European Parliament, (1985). In: *Committee of Inquiry into the Rise of Fascism and Racism in Europe*. [online] p.18. Available at: <https://op.europa.eu/en/publication-detail/-/publication/557b5ea7-34f9-4399-aa0d-14f19aab1d90> [Accessed 1 July 2020].

Fielitz, M. and Marcks. H. (2019). Digital Fascism: Challenges for the Open Society in Times of Social Media. In: *UC Berkeley: Center for Right-Wing Studies*. [online]. Available at: <https://escholarship.org/uc/item/87w5c5gp> [Accessed 17 June 2020].

Finchelstein, F., (2017). *From Fascism To Populism In History*. 1st ed. [ebook] Oakland, California: University of California Press, pp.31-97. Available at: <https://www.jstor.org/stable/j.ctvpb3vkk> [Accessed 21 June 2020].

Foster, J. (2017). *Trump in the White House: Tragedy and Farce.* [ebook] New York: NYU Press, pp.19-56. Available at: <https://www.jstor.org/stable/j.ctt1qv5qzf> [Accessed 28 June 2020].

Gambäck, B. and Sikdar, U., (2017). Using Convolutional Neural Networks to Classify Hate-Speech. *Proceedings of the First Workshop on Abusive Language Online*, [online] Available at: <https://www.aclweb.org/anthology/W17-3013/> [Accessed 9 July 2020].

Géron, A., (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, And Tensorflow*. Sebastopol: O'Reilly Media, Incorporated.
Molnar, C., (2020). *Interpretable Machine Learning*. 1st ed. Leanpub, pp.71-83.

Gitari, N., Zhang, Z., Damien, H. and Long, J., (2015). A Lexicon-based Approach for Hate Speech Detection. *International Journal of Multimedia and Ubiquitous Engineering*, [online] 10(4), pp.215-230. Available at: <http://gvpress.com/journals/IJMUE/vol10_no4/21.pdf> [Accessed 28 June 2020].

Go, A., Bhayani, R. and Huang, L., (2009). Twitter Sentiment Classification using Distant Supervision. [online] Available at: <https://www-cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf> [Accessed 4 July 2020].

Griffin, R., (1991). *The Nature of Fascism*. 1st ed. New York: St. Martin's Press, pp.26-44.

Griffin, R. (2000). Interregnum or endgame? The radical right in the 'post-fascist' era. *Journal of Political Ideologies*, [online] Volume 5(2), pp.163-178. Available at: <https://doi.org/10.1080/713682938> [Accessed 15 June 2020].

Gröndahl, T., Pajola, L., Juuti, M., Conti, M. and Asokan, N., (2018). All You Need is. *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security - AISec '18*, [online] Available at: <https://doi.org/10.1145/3270101.3270103> [Accessed 18 July 2020].

Hawley, G. (2017). *Making Sense of the Alt-Right.* [ebook] New York: Columbia University Press, pp.11-50. Available at: <https://www.jstor.org/stable/10.7312/hawl18512> [Accessed 17 June 2020].

Indurthi, V., Syed, B., Shrivastava, M., Chakravartula, N., Gupta, M. and Varma, V., (2019). FERMI at SemEval-2019 Task 5: Using Sentence embeddings to Identify Hate Speech Against Immigrants and Women in Twitter. *Proceedings of the 13th International Workshop on Semantic Evaluation*, [online] Available at: <https://www.aclweb.org/anthology/S19-2009> [Accessed 15 July 2020].

James, G., Witten, D., Hastie, T. and Tibshirani, R., (2013). *An Introduction to Statistical Learning with Applications in R*. [ebook] New York: Springer. Available at: <https://link.springer.com/book/10.1007/978-1-4614-7138-7> [Accessed 24 July 2020].

Jones, R., (2014). *The Fascist Party In Wales? : Plaid Cymru, Welsh Nationalism And The Accusation Of Fascism*. [ebook] University of Wales Press, pp.31-48. Available at: <https://ebookcentral.proquest.com/lib/bham/detail.action?docID=1889171> [Accessed 25 June 2020].

Kanaris, I., Kanaris, K., Houvardas, I. and Stamatos, E., (2007). Words Versus Character n-grams for Anti-spam Filtering. *International Journal on Artificial Intelligence Tools*, [online] 16(06), pp.1047-1067. Available at: <http://www.icsd.aegean.gr/lecturers/Stamatatos/papers/IJAIT-spam.pdf> [Accessed 30 June 2020].

Kotsiantis, S., (2007). Supervised Machine Learning: A Review of Classification Techniques. [online] pp.249-268. Available at: <http://www.informatica.si/index.php/informatica/article/viewFile/148/140> [Accessed 13 August 2020].

Kwok, I. and Wang, Y., (2013). Locate the Hate: Detecting Tweets against Blacks. In: P*roceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*. [online] pp.1621–1622. Available at: <https://dl.acm.org/doi/10.5555/2891460.2891697> [Accessed 23 July 2020].

Magnan, S., (2019). *1 Million Reddit Comments From 40 Subreddits*. [online] Kaggle.com. Available at: <https://www.kaggle.com/smagnan/1-million-reddit-comments-from-40-subreddits> [Accessed 12 June 2020].

Malmasi, S. and Zampieri, M., (2017). Detecting Hate Speech in Social Media. In: *Proceedings of Recent Advances in Natural Language Processing (RANLP)*. [online] pp.467-472. Available at: <https://arxiv.org/abs/1712.06427> [Accessed 5 July 2020].

Malmasi, S. and Zampieri, M., (2018). Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, [online] 30(2), pp.187-202. Available at: <https://arxiv.org/abs/1803.05495> [Accessed 6 July 2020].

Mikolov, T., Chen, K., Corrado, G. and Dean, J., (2013). Efficient Estimation of Word Representations in Vector Space. In: *Conference: Proceedings of the International Conference on*

Learning Representations (ICLR 2013). [online] Available at: <https://arxiv.org/abs/1301.3781> [Accessed 2 July 2020].

Moschitti, A. and Basili, R., (2004). Complex Linguistic Features for Text Classification: A Comprehensive Study. In: *McDonald S., Tait J. (eds) Advances in Information Retrieval. ECIR 2004. Lecture Notes in Computer Science, vol 2997*. [online] Available at: <https://doi.org/10.1007/978-3-540-24752-4_14> [Accessed 8 July 2020].

Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y. and Chang, Y., (2016). Abusive Language Detection in Online User Content. *Proceedings of the 25th International Conference on World Wide Web - WWW '16*, [online] Available at: <https://doi.org/10.1145/2872427.2883062> [Accessed 6 July 2020].

Le, Q. and Mikolov, T., (2014). Distributed Representations of Sentences and Documents. In: *Proceedings of the 31st International Conference on Machine Learning, PMLR 32(2)*. [online] pp.1188 -1196. Available at: <http://proceedings.mlr.press/v32/le14.html> [Accessed 7 July 2020].

O'Brien, L., (2018). *American Neo-Nazi Is Using Holocaust Denial as A Legal Defence*. [online] The Huffington Post. Available at: <https://www.huffingtonpost.co.uk/entry/neo-nazi-holocaust-denial-legal-defense_n_5a612a5ce4b0125fd6354368?ri18n> [Accessed 30 June 2020].

Passmore, K., (2014). *Fascism: A Very Short Introduction*. 2nd ed. [ebook] Oxford University Press, pp.1-20. Available at: <https://www.veryshortintroductions.com/view/10.1093/actrade/9780199685363.001.0001/actrade-9780199685363> [Accessed 31 June 2020].

Pauley, B., 2014. *Hitler, Stalin, And Mussolini: Totalitarianism In The Twentieth Century*. 4th ed. [ebook] West Sussex: John Wiley & Sons, Incorporated, pp.223-265. Available at: <https://ebookcentral.proquest.com/lib/bham/detail.action?docID=1729068> [Accessed 26 June 2020].

Payne, S., (1983). *Fascism: Comparison and Definition*. [ebook] Wisconsin: University of Wisconsin Press, pp.3-21. Available at: <https://ebookcentral.proquest.com/lib/bham/detail.action?docID=3445188> [Accessed 29 June 2020].

Pham, D., (2019). *Iron March 'Fascist Social Network' Dataset*. [online] Kaggle.com. Available at: <https://www.kaggle.com/davidpham/iron-march-fascist-social-network-dataset> [Accessed 11 June 2020].

Porter, M., (2001). Snowball: A language for stemming algorithms. [online] Available at: <https://pdfs.semanticscholar.org/0d8f/907bb0180912d1e1df279739e45dff6853ee.pdf> [Accessed 30 July 2020].

Portilla, J., (2019). *Natural Language Processing with Python*. [online] Udemy. Available at: <https://www.udemy.com/course/nlp-natural-language-processing-with-python/> [Accessed 17 June 2020].

Qian, J., Bethke, A., Liu, Y., Belding, E. and Wang, W., (2019). A Benchmark Dataset for Learning to Intervene in Online Hate Speech. [online] Available at: <https://arxiv.org/abs/1909.04251> [Accessed 10 August 2020].

Renton, D., (1999). *Fascism: Theory and Practice*. [ebook] London; Sterling, Virginia: Pluto Press, pp.6-17. Available at: <https://www.jstor.org/stable/j.ctt18dzsj6> [Accessed 28 June 2020].

Samuel, A., (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, [online] 3(3), pp.210-229. Available at: <http://www2.stat.duke.edu/~sayan/R_stuff/Datamatters.key/Data/samuel_1959_B-95.pdf> [Accessed 18 August 2020].

Schmidt, A. and Wiegand, M., (2017). A Survey on Hate Speech Detection using Natural Language Processing. *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, [online] Available at: <https://www.aclweb.org/anthology/W17-1101> [Accessed 2 July 2020].

scikit-learn, (2020). *Cross Validation*. [image] Available at: <https://scikit-learn.org/stable/modules/cross_validation.html> [Accessed 11 August 2020].

Shennan, R., (2020). *What Did Wiley Say On Twitter? Antisemitic Tweets from The Grime Rapper Explained As He Is Banned Permanently From Social Media Platforms*. [online] Scotsman.com. Available at: <https://www.scotsman.com/whats-on/arts-and-entertainment/what-did-wiley-say-twitter-antisemitic-tweets-grime-rapper-explained-he-banned-permanently-social-media-platforms-2925134> [Accessed 23 August 2020].

Timberg, C. and Dwoskin, E., (2020). *Reddit Closes Long-Running Forum Supporting President Trump After Years of Policy Violations*. [online] The Washington Post. Available at: <https://www.washingtonpost.com/technology/2020/06/29/reddit-closes-long-running-forum-supporting-president-trump-after-years-policy-violations/> [Accessed 2 July 2020].

Tjihero, N., (2018). *Data Augmentation For Text Data: Obtain More Data Faster*. [online] Towards Data Science. Available at: <https://towardsdatascience.com/data-augmentation-for-text-data-obtain-more-data-faster-525f7957acc9> [Accessed 4 July 2020].

Tondo, L., (2019). *Facebook Closes Italian Neo-Fascist Party's Account*. [online] The Guardian. Available at: <https://www.theguardian.com/world/2019/sep/09/facebook-closes-italian-neo-fascist-party-casapound-account> [Accessed 30 June 2020].

Traverso, T. (2019). *The New Faces of Fascism: Populism and the Far Right*. English-language ed. Brooklyn: Verso, pp.3-19.

Tsukayama, H., (2018). *Facebook Turns To Artificial Intelligence To Fight Hate And Misinformation In Myanmar*. [online] The Washington Post. Available at: <https://www.washingtonpost.com/technology/2018/08/16/facebook-turns-artificial-intelligence-fight-hate-misinformation-myanmar/> [Accessed 27 June 2020].

Warner, W. and Hirschberg, J., (2012). Detecting Hate Speech on the World Wide Web. In: *Proceedings of the Second Workshop on Language in Social Media*. [online] Association for Computational Linguistics, pp.19-26. Available at: <https://www.aclweb.org/anthology/W12-2103> [Accessed 30 June 2020].

Waseem, Z. and Hovy, D., (2016). Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. *Proceedings of the NAACL Student Research Workshop*, [online] Available at: <https://www.aclweb.org/anthology/N16-2013/> [Accessed 3 July 2020].

Wei, J. and Zou, K., (2019). EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. [online] pp.6383 - 6389. Available at: <https://www.aclweb.org/anthology/D19-1670> [Accessed 6 July 2020].

Willie Thompson, W., (2011). *Ideologies in The Age Of Extremes: Liberalism, Conservatism, Communism, Fascism 1914-1991*. [ebook] London; New York: Pluto Press, pp.85-102. Available at: <https://www.jstor.org/stable/j.ctt183pcxs> [Accessed 25 June 2020].

Wilson, J., (2019). *Leak from Neo-Nazi Site Could Identify Hundreds of Extremists Worldwide*. [online] The Guardian. Available at: <https://www.theguardian.com/us-news/2019/nov/07/neo-nazi-site-iron-march-materials-leak> [Accessed 10 July 2020].

Xu, R., Chen, T., Xia, Y., Lu, Q., Liu, B. and Wang, X., (2015). Word Embedding Composition for Data Imbalances in Sentiment and Emotion Classification. *Cognitive Computation*, [online] 7(2), pp.226-240. Available at: <https://doi.org/10.1007/s12559-015-9319-y> [Accessed 22 July 2020].

Appendices

Appendix A: GitLab Repository and Source Code Information

All source code created for this project can be found in the GitLab repository located at the following URL: https://git-teaching.cs.bham.ac.uk/mod-msc-proj-2019/sxd942

The folder 'Code' contains two directories. The first, 'Dataset_creation', contains four IPYNB files that were written in a Jupyter notebook and relate to the creation of the csv files for the original four datasets used for binary classification. The second, 'classification2', contains the Python scripts and datasets that comprise the classification test system architecture that was detailed in section 6.6.

Code Instructions

To recreate the classification experiments that were performed in this research, download the directory from the repository as a Zip file and extract the directory 'classification2' to your desired location. Inside your Python IDE of preference, open this directory and navigate to the package titled 'classification'. Inside this package you will see the script named 'classification_experiments.py' which contains two methods:

1.) To run the binary classification experiments:

I. Uncomment the method call: *make_classification(feature, model, grid_search)*
II. Referring to either the documentation in the script or the README.md file replace the highlighted with your chosen parameters and run the script.

2.) To run the multiclass classification experiments:

I. Uncomment the method call: *make_multi_classification(feature, model, grid_search, both)*
II. Referring to either the documentation in the script or the README.md file replace the highlighted with your chosen parameters and run the script.

Additional Scripts

Additional scripts that can be run from inside the 'classification2' directory include:

I. The package 'data_visualisation' contains four Python scripts, each can be run individually to recreate the visual figures created for the project.
II. The package 'classification' contains a script titled 'model_analysis.py' which can be run to view the results of the Tweet analysis.

Code Dependencies and References

In the README.md file:
Section 4 provides a list of Python modules that are required to be installed to run the code.
Section 5 provides a list of code references that were used for this project.

Appendix B: Binary Classification Reports

The following tables are the classification reports relating to the experiments conducted in chapter 7, 'Binary Classification: Fascist vs. Non-fascist speech' (see 7.2).

		Fascist vs Non-Fascist Speech: **Gold Dataset**									
		Fascist			Non-fascist			Weighted-Average			Acc.
		P	R	F1	P	R	F1	P	R	F1	A
Linear-SVC	Tf-idf word n-grams	0.96	0.90	0.93	0.93	0.97	0.95	0.94	0.94	0.94	0.94
	Tf-idf char n-grams	0.94	0.95	**0.95**	0.96	0.96	0.96	0.95	0.95	0.95	0.95
	Word embeddings	0.83	0.89	0.86	0.91	0.86	0.88	0.87	0.87	0.87	0.87
	Paragraph Vector	0.88	0.75	0.81	0.85	0.93	0.89	0.86	0.86	0.86	0.86
Logistic Regression	Tf-idf word n-grams	0.95	0.91	0.93	0.94	0.97	0.95	0.94	0.94	0.94	0.94
	Tf-idf char n-grams	0.94	0.93	0.93	0.95	0.96	0.95	0.95	0.95	0.95	0.94
	Word embeddings	0.86	0.84	0.85	0.88	0.90	0.89	0.87	0.87	0.87	0.87
	Paragraph Vector	0.88	0.79	0.83	0.87	0.93	0.90	0.87	0.87	0.87	0.87
Random Forest	Tf-idf word n-grams	0.91	0.90	0.90	0.93	0.94	0.94	0.92	0.92	0.92	0.92
	Tf-idf char n-grams	0.90	0.88	0.89	0.92	0.93	0.93	0.91	0.91	0.91	0.91
	Word embeddings	0.94	0.84	0.89	0.89	0.96	0.92	0.91	0.91	0.91	0.91
	Paragraph Vector	0.85	0.74	0.79	0.84	0.91	0.87	0.84	0.84	0.84	0.84

Table 13: Classification report for *Gold* dataset.

		Fascist vs Non-Fascist Speech: **Shuffled Dataset**									
		Fascist			Non-fascist			Weighted-Average			Acc.
		P	R	F1	P	R	F1	P	R	F1	A
Linear-SVC	Tf-idf word n-grams	0.97	0.93	**0.95**	0.97	0.99	0.98	0.97	0.97	0.97	0.97
	Tf-idf char n-grams	0.93	0.94	0.93	0.98	0.98	0.98	0.97	0.97	0.97	0.97
	Word embeddings	0.74	0.97	0.84	0.99	0.89	0.93	0.93	0.91	0.91	0.91
	Paragraph Vector	0.69	0.74	0.71	0.91	0.89	0.90	0.85	0.85	0.85	0.85
Logistic Regression	Tf-idf word n-grams	0.91	0.96	0.93	0.99	0.97	0.98	0.97	0.96	0.97	0.97
	Tf-idf char n-grams	0.82	0.94	0.88	0.98	0.93	0.95	0.94	0.93	0.93	0.93
	Word embeddings	0.76	0.96	0.85	0.99	0.90	0.94	0.93	0.91	0.91	0.91
	Paragraph Vector	0.66	0.73	0.69	0.91	0.88	0.89	0.84	0.84	0.84	0.84
Random Forest	Tf-idf word n-grams	0.97	0.50	0.66	0.85	1.00	0.92	0.88	0.87	0.85	0.87
	Tf-idf char n-grams	0.86	0.91	0.88	0.97	0.95	0.96	0.94	0.94	0.94	0.94
	Word embeddings	0.93	0.83	0.88	0.95	0.98	0.96	0.94	0.94	0.94	0.94
	Paragraph Vector	0.69	0.84	0.76	0.94	0.87	0.91	0.88	0.86	0.87	0.87

Table 14: Classification report for *Shuffled* dataset.

		Fascist vs Non-Fascist Speech: **SR (Synonym Replacement) Dataset**									
		Fascist			Non-fascist			Weighted-Average			Acc.
		P	R	F1	P	R	F1	P	R	F1	A
Linear-SVC	Tf-idf word n-grams	0.88	0.82	0.85	0.96	0.98	0.97	0.95	0.95	0.95	0.94
	Tf-idf char n-grams	0.88	0.89	0.89	0.98	0.97	0.98	0.96	0.96	0.96	0.95
	Word embeddings	0.58	0.84	0.69	0.96	0.87	0.91	0.89	0.86	0.87	0.86
	Paragraph Vector	0.60	0.76	0.67	0.94	0.89	0.92	0.88	0.87	0.87	0.86
Logistic Regression	Tf-idf word n-grams	0.82	0.86	0.84	0.97	0.96	0.96	0.94	0.94	0.94	0.94
	Tf-idf char n-grams	0.75	0.91	0.82	0.98	0.93	0.96	0.94	0.93	0.93	0.93
	Word embeddings	0.57	0.85	0.68	0.96	0.86	0.91	0.89	0.86	0.87	0.85
	Paragraph Vector	0.56	0.74	0.64	0.94	0.87	0.90	0.87	0.85	0.86	0.86
Random Forest	Tf-idf word n-grams	0.95	0.64	0.77	0.93	0.99	0.96	0.93	0.93	0.92	0.93
	Tf-idf char n-grams	0.69	0.85	0.76	0.97	0.92	0.94	0.92	0.91	0.91	0.90
	Word embeddings	0.78	0.81	0.80	0.96	0.95	0.95	0.93	0.93	0.93	0.93
	Paragraph Vector	0.55	0.78	0.65	0.95	0.86	0.90	0.88	0.85	0.86	0.86

Table 15: Classification report for *SR* (*Synonym Replacement*) dataset.

		Fascist vs Non-Fascist Speech: **Synthetic Dataset**									
		Fascist			Non-fascist			Weighted-Average			Acc.
		P	R	F1	P	R	F1	P	R	F1	A
Linear-SVC	Tf-idf word n-grams	1.00	0.62	0.77	0.73	1.00	0.84	0.86	0.81	0.81	0.81
	Tf-idf char n-grams	1.00	0.59	0.74	0.71	1.00	0.83	0.85	0.80	0.79	0.80
	Word embeddings	0.93	0.89	**0.91**	0.89	0.93	0.91	0.91	0.91	0.91	0.91
	Paragraph Vector	0.96	0.85	**0.90**	0.86	0.97	0.91	0.91	0.91	0.91	0.91
Logistic Regression	Tf-idf word n-grams	0.99	0.76	0.86	0.80	0.99	0.89	0.90	0.88	0.87	0.88
	Tf-idf char n-grams	1.00	0.73	0.84	0.79	1.00	0.88	0.89	0.87	0.86	0.86
	Word embeddings	0.93	0.88	0.90	0.89	0.93	0.91	0.91	0.91	0.91	0.91
	Paragraph Vector	0.94	0.80	0.87	0.83	0.95	0.88	0.89	0.88	0.88	0.88
Random Forest	Tf-idf word n-grams	1.00	0.10	0.18	0.53	1.00	0.69	0.76	0.55	0.44	0.55
	Tf-idf char n-grams	1.00	0.28	0.43	0.58	1.00	0.73	0.79	0.64	0.58	0.64
	Word embeddings	1.00	0.46	0.63	0.65	1.00	0.79	0.82	0.73	0.71	0.73
	Paragraph Vector	0.97	0.49	0.65	0.66	0.99	0.79	0.82	0.74	0.72	0.74

Table 16: Classification report for *Synthetic* dataset.