



University of East London

Research Dissertation

Sentiment Analysis - Detection of Abusive Language on Social Networks using Machine Learning

VASILEIOS TILLIS

UNIVERSITY OF EAST LONDON

Msc Data Science

Student ID:

2121807

August, 2021

Abstract

The objective of this thesis is to create models that can identify abusive language in social media platforms such as Twitter Data Set and Detecting Insults in Social Commentary. Two models were developed for this purpose, and both were trained and evaluated on Twitter data (Wassem and Hovy, 2016). Initially, we applied the Naïve Bayes model as it is the most commonly used technique in text classification and evaluation (Wei Zhang and Feng Geo, 2011). Subsequently, we used the RNN (Recurrent Neural Networks), which is a more efficient method for text classification compared to Naïve Bayes (Maaz Amaid, Zhanibek Kaimuldenov, Ilia Voronkov, 2016). To test the accuracy and effectiveness of the models, we evaluated them on another dataset, i.e., Detecting Insults in Social Commentary. Finally, we analyzed and adjusted the results of the models accordingly.

Table of contents

<i>Abstract</i>	2
<i>Module 1</i>	4
1.1 <i>Introduction</i>	4
1.2 <i>The drive behind the dissertation</i>	4
1.3 <i>Goals and Objectives</i>	5
1.4 <i>Brief overview</i>	5
<i>Module 2</i>	6
2.1 <i>Review of the Bibliography</i>	6
2.2 <i>Analysis of the methods used</i>	12
<i>Module 3</i>	14
3.1 <i>Datasets</i>	14
3.2 <i>Naïve Bayes model</i>	14
3.2.1 <i>Naïve Bayes train/test set</i>	17
3.2.2 <i>Application of Naïve Bayes on new data</i>	19
3.3 <i>Recurrent Neural Networks</i>	22
3.3.1 <i>RNN train/test set</i>	26
3.3.2 <i>Application of RNN model on new data</i>	28
<i>Module 4</i>	33
4.1 <i>Discussion of results</i>	33
4.2 <i>Suggestions for future work</i>	35
<i>Module 5</i>	37
<i>Bibliography</i>	37

Module 1

1.1 Introduction

In recent times, there has been a significant surge in the usage of offensive language on the internet, which has prompted more people to seek effective ways to combat it. Online platforms such as forums and social media have unfortunately become hubs for bullying and hate speech. These forms of online aggression not only create a toxic social environment but also have the potential to cause harm, both mental and physical. This thesis focuses on analyzing the sentiment of several tweets to identify abusive language on Twitter. Sentiment analysis is an important tool for researchers in fields like human-computer interaction, sociology, marketing, advertising, psychology, economics, and political science. However, social media content, such as that found on Twitter or Facebook, poses significant challenges to practical applications of sentiment analysis due to the sheer volume of user-generated content, which often uses abbreviated linguistic conventions that are typically keywords for sentiment analysis.

To extract emotions from tweets for this project, a combination of natural language processing (NLP) and machine learning techniques were employed. NLP, which deals with computational language analysis, provides effective tools for developing automated methods to analyze, detect, and filter abusive language. Meanwhile, machine learning algorithms were utilized to enable the program to learn from past textual data and anticipate the polarity of abusive language, whether it is abusive or non-abusive.

1.2 The drive behind the dissertation

The amount of social data being generated is increasing at a rapid pace. According to Global WebIndex statistics, there has been a 17% annual rise in mobile users, with a total of 3.7 billion unique users utilizing mobile phones (D. Chaffey, 2016). Social networking sites have become an established platform for users to express their thoughts and opinions on a variety of topics such as events, individuals, or products. Social media channels have also become a popular means for people to discuss ideas and communicate with individuals worldwide. Facebook, for example, claims to have 1.59 billion monthly active users, with the average user being friends with 130 people (W. Tan, M.B. Blake, I. Saleh, S. Dustdar, 2013). Similarly, Twitter boasts over 500 million users, with more than 332 million being active (D. Chaffey, 2016). Users post over 340 million tweets and 1.6 billion search queries each day. However, this massive influx of data presents numerous challenges. When an event occurs or a product is launched, people start tweeting, writing reviews, and posting comments on social media to get the opinions of other users before making a purchase. Organizations also rely on these sites to gather user feedback about their products and use it to improve them. However, identifying and verifying the legitimacy of these opinions and reviews is a daunting task, requiring a lot of time and effort to manually read through everything and determine the sentiment expressed. As a result, it is essential to use tools that can simplify the entire process. Natural language processing and machine learning algorithms can solve this problem, with NLP providing a tool for analyzing large amounts of data and ML algorithms being used to classify it appropriately.

1.3 Goals and Objectives

The objective of this study is to create two models that can identify abusive language on social networks. To accomplish this, the study utilized two popular techniques in sentiment classification: the Naïve Bayes model and the recurrent neural network model (RNN). The study utilized Twitter data (such as `amateur_expert.json`) for training and evaluating the models. The data was extracted using natural language processing, and machine learning algorithms were used for tweet classification. The Naïve Bayes model is a simple model that employs probabilities to generate results, while the recurrent neural network model utilizes artificial neural networks with replay learning. Both models were applied to the same dataset, and their accuracy was evaluated on different data (such as Detecting on Insults in Social Commentary, <https://www.kaggle.com/c/detecting-insults-in-social-commentary/data>) to enable comparison and further analysis

1.4 Brief overview

The thesis is structured into five main sections. In the first section, the aims and objectives of the research are introduced, while in the second section, related techniques used by other researchers for emotion analysis are discussed. The third section provides a detailed description of all the tools and techniques that were employed in developing the models, whereas the fourth section delves into the results of the models, along with potential future applications. Finally, the last section comprises a list of the literature references cited in this thesis..

Module 2

2.1 Review of the Bibliography

Introduction

Hate speech is a prevalent issue on the Internet (Eadicicco, 2014; Kettrey and Laster, 2014) and can have serious consequences for the individuals who are subjected to it. Therefore, detecting it is crucial in preventing hate crimes by targeting the words and phrases in posts that can be identified on the internet (Watch, 2014). Social media platforms such as Twitter and Facebook have become a popular outlet for users to express their opinions and feelings anonymously, resulting in an increasing number of researchers focusing on these platforms for detecting abusive language through sentiment analysis.

Sentiment analysis is characterized by the identification or extraction of the content of a text module using NLP, statistics, or machine learning methods. This paper aims to detect abusive language on Twitter using NLP and machine learning methods and refers to previous studies that have focused on this topic and how other researchers have approached abusive language detection using sentiment analysis.

One such study was conducted by Waseem and Hovy (2016), who developed an abusive language detection model by evaluating the effect of different features on prediction in a classification task using the logistic regression model and 10-fold cross-validation to test the performance of their prediction. The researchers collected Twitter data and manually classified it into three categories: sexism, racism, and neither. However, for their research, they only used the first two categories to train their model.

The researchers trained the model on Twitter comments using different attributes each time. Initially, they trained the model using bi- to four-grams characters along with gender information, and then they used the user's gender and location along with 1- to 4-grams characters. Wasseem and Hovy found that the last experiment, which included more information, produced the best result. The experiments conducted by the researchers are detailed in the following table.

	char n-grams	+gender	+gender +loc	word n-grams
F1	73.89	73.93	73.62*	64.58
Precision	72.87%	72.93%	72.58%	64.39%
Recall	77.75%	77.74%	77.43%	71.93%

Table 1: Results n-grams using different characteristics

It should be noted that the algorithm discussed is a binary classification method, which means it can only be used with two categories and cannot be extended to three categories (neither, sexism, racism).

In another study on identifying abusive language using sentiment analysis, Bourgonje, Moreno-Scheider, and Ankit Srivastan (2018) found that there was no need for automatic categorization of online content, except in two cases: online advertisements and sentiment analysis of social network data. In these cases, categorization into specific categories such as "hate language," "abusive language," or "false news" is necessary. However, categorizing abusive language accurately remains a challenge due to the difficulties posed by irony, sarcasm, and other issues.

These researchers focused on the application of several categorization algorithms to test the ability to detect and categorize abusive language. They used publicly available data and six classifiers: Bayes, Bayes expectation maximization, C4.5 decision trees, multivariate logistic regression, maximum entropy, and Winnow2. The Mallet Toolkit for Machine Learning Language (McCallum 2002) was used for all three datasets. The table below shows the scores of the classifiers.

	Bayes	Bayes exp. max.	C4.5	Logistic Regression	Maximum Entropy	Winnow2		Bayes	Bayes exp. max.	C4.5	Logistic Regression	Maximum Entropy	Winnow2
	English Tweets (ET)							German Tweets (GT) – (binary, exp. 1)					
accuracy	84.61	84.01	82.95	85.67	83.67	76.66	accuracy	75.74	78.93	74.04	77.23	75.96	71.91
precision	80.54	79.57	79.07	83.57	81.20	69.85	precision	70.65	75.07	69.30	74.80	72.46	72.41
recall	78.63	77.97	74.37	77.45	74.37	69.62	recall	74.78	76.06	74.98	76.58	74.85	72.68
f-score	79.10	78.34	76.17	80.06	77.20	69.32	f-score	65.84	69.74	70.66	71.98	73.02	71.15
	German Tweets (GT) – (binary, exp. 2)							German Tweets (GT) – (rating)					
accuracy	80.21	74.26	76.81	79.15	76.38	77.23	accuracy	36.60	35.32	37.87	33.40	34.89	25.53
precision	72.76	73.59	72.54	77.18	73.62	74.65	precision	42.51	39.76	56.22	31.39	31.90	38.17
recall	77.57	79.49	77.85	79.74	77.31	76.37	recall	38.53	38.19	38.76	36.34	35.71	25.84
f-score	70.93	68.97	69.85	75.41	74.20	73.05	f-score	27.43	27.03	23.68	30.34	30.75	24.06
	Wikipedia (WT) – Attack (binary)							Wikipedia (WT) – Aggression (binary)					
accuracy	83.11	82.70	81.08*	80.90	77.71	77.77	accuracy	82.19	82.10	79.58*	80.42	77.17	79.08
precision	81.78	81.33	79.27*	79.36	76.03	77.11	precision	80.68	80.60	78.13*	78.91	75.26	77.25
recall	83.14	82.83	81.31*	80.97	77.87	77.83	recall	82.01	81.87	80.18*	80.46	77.29	78.57
f-score	81.58	81.36	79.27*	79.74	76.65	77.28	f-score	80.60	80.57	78.37*	79.23	75.80	77.45
	Wikipedia (WT) – Aggression (rating)												
accuracy	67.13	67.40	66.81*	65.28	57.77	55.73							
precision	57.21	56.05	54.08*	57.42	57.21	54.07							
recall	67.27	66.94	66.42*	65.68	58.18	55.73							
f-score	59.13	59.00	58.14*	59.95	55.26	54.53							

Figure 1: Results of categorisation experiments

Bourgonje, Moreno-Scheider, and Ankit Srivastan conducted categorization experiments using multiple algorithms on three datasets that differed in language, size, and topic.

The datasets included a corpus of English tweets tagged for racism and sexism (Wassem and Hovy, 2016), a corpus of German tweets flagged for "hate language" (Ross et al., 2016), and a corpus of English Wikipedia user comments with fewer size restrictions (Wulezyn et al., 2016).

The researchers analyzed the performance differences between the various algorithms and identified the optimal algorithm for each dataset (Figure 1). The results indicate that automatic categorization can effectively detect abusive language, and it is noteworthy that all classifiers used a simple BOW set, which could potentially be replaced by more conceptually and thematically adapted elements like value-based emotion attribution. The datasets were also cleaned of colloquial expressions, a process that could be improved with better natural language processing (NLP) techniques to enhance accuracy.

Previous research studies utilized traditional algorithms that have been in use for decades. However, Gamback and Sikdar (2017) introduced a novel approach using deep learning-based text categorization system for classifying hate speech text on Twitter. They utilized convolutional neural networks (CNN) that have been proven to be effective in solving various language processing tasks such as sentiment analysis, part-of-speech tagging, and ontological recognition. The researchers constructed two CNN models that map tweets to one of the four categories: racism, sexism, both, or neither. The models were trained and categorized based on different sets of input vectors, including verbal vectors based on semantic information generated using either a non-constrained strategy or word2vec, as well as character 4-grams and a combination of verbal vectors and character n-grams. The max-pooling layer reduced the set of interest, and the SoftMax layer mapped tweets to their most likely category. The classifier underwent the 10-fold cross-validation procedure for training and testing. The best score of 78.3% was obtained using the word2vec word vector set. Table 1 presents the results of the CNN model in comparison with another classifier, linear regression with character n-grams (Wassem and Hovy, 2016).

System setup	Precision	Recall	F1-score
Random vectors	0.8668	0.6726	0.7563
word2vec	0.8566	0.7214	0.7829
Character n-grams	0.8557	0.7011	0.7695
word2vec + character n-grams	0.8661	0.7042	0.7738
Logistic Regression with character n-grams (Waseem and Hovy, 2016)	0.7287	0.7775	0.7389

Table 2: System performance (10-fold cross validated)

Ji Ho Park and Pascale Fung (2017) conducted a study that employed various CNN classifiers to detect "abusive language". Their two-step approach involved first detecting "abusive language" and then categorizing it into different subtypes. A comparison was also made between this approach and a one-step method where categorization into several classes of sexist and racist

language was carried out. For categorization in different separated datasets, the researchers used three types of CNN models with character-level but word-level input vectors.

These models were CharCNN, WordCNN, and HybridCNN and differed based on whether the input features were characters, words, or hybrids. Two datasets of English tweets were used, but were segmented for multiclass categorization in the one-step method. For the two-step categorization, tweets of racist and sexist content were combined into one category, "abusive language". Finally, a third dataset of abusive language was created for experimentation with an additional categorizer to test its ability to distinguish between sexist and racist content if a tweet was categorized as abusive language. Table 2 shows the trimmed dataset.

Dataset	One-Step			Two-step-1		Two-Step-2	
Label	None	Racism	Sexism	None	Abusive	Sexism	Racism
#	12,427	2,059	3,864	12,427	5,923	2,059	3,864

Table 3: Trimmed dataset

The objective of the experiments was to determine if breaking down the problem into two steps would be more effective. In addition, the results were compared to those of other classifiers that were previously used by other researchers, such as linear regression (Wassem and Hovy, 2016), FastText (Joulin et al., 2016), and CNN (Badjatiya et al., 2017). The outcomes indicate that the HybridCNN model for multiclass categorization in one step is more effective than the WordCNN model. While the other classifiers tend to have a higher average F1-score, they perform poorly for individual categories of racism and sexism. The two-step approach, which combines two binary classifiers, achieved similar results to the one-step approach. Additionally, combining two logistic regression classifiers in the two-step approach performed as well as the one-step approach. The results of both approaches are presented in Figure 2.

Method	None			Racism			Sexism			Total		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
LR	.824	.945	.881	.810	.598	.687	.835	.556	.668	.825	.824	.814
SVM	.802	.956	.872	.815	.531	.643	.851	.483	.616	.814	.808	.793
FastText	.828	.922	.882	.759	.630	.685	.777	.557	.648	.810	.812	.804
CharCNN	.861	.867	.864	.693	.746	.718	.713	.666	.688	.801	.811	.811
WordCNN	.870	.868	.868	.704	.762	.731	.712	.686	.694	.818	.816	.816
HybridCNN	.872	.882	.877	.713	.766	.736	.743	.679	.709	.827	.827	.827
LR (two)	.841	.933	.895	.800	.664	.731	.809	.590	.683	.828	.831	.824
SVM (two)	.816	.945	.876	.811	.605	.689	.823	.511	.630	.816	.815	.803
HybridCNN (two)	.877	.864	.869	.690	.759	.721	.705	.701	.699	.807	.809	.807
HybridCNN + LR(two)	.880	.859	.869	.722	.751	.735	.683	.717	.699	.821	.817	.818

Figure 2 The results of the experiment show two parts: the upper part represents the first step of the method, which involves multi-class classification, while the lower part represents the two-step approach that combines two binary classifiers. The new model introduced in this study is called HybridCNN.

The experiments presented here used the convolutional network architecture, but other deep neural networks could also be exploited. For example, the Long Short-Term Memory (LSTM) recurrent neural network (recurrent neural network or RNN) seems useful for language processing problems in which exploiting the sequential nature of the input information is more important, as in the case of entity recognition and sentiment analysis. Theodora Chu, Kylie Jue, Max Wang (2017) used 3 different models to categorize personal and non-personal attacks. The researchers applied three classifiers to the Wikipedia user interaction page: long-short term memory cell (LSTM) RNN with lexical embeddings, CNN with lexical embeddings, CNN with character embeddings. Using deep learning models they tried to see if a comment could be clarified whether it should be managed or not. So they first used two lines of reference to understand the problem from both the machine learning and deep learning perspectives. So they used linear regression with a least squares loss function to generate the machine learning baseline and a neural network classifier for the deep learning baseline. Table 3 shows the results of the reference lines.

Dataset	F1 Score	Accuracy	AUC
Linear regression	0.45	0.91	0.89
Deep neural network	0.54	0.91	0.91

Table 4: Baseline results

The results of the experiments demonstrate that the deep neural network outperformed the linear regression model, achieving F1-scores of 0.54 and 0.45, respectively. This suggests that a neural network is more effective than a conventional machine learning model. The subsequent phase involved training these deep learning algorithms. The CNN model using character

embeddings required a larger number of training steps (50000) to attain an F1-score of 0.73. In contrast, the CNN model using lexical embeddings reached an F1-score of 0.70 after only 5000 training steps, with no further improvement observed upon additional training. The LSTM model showed similar performance to the CNN model with lexical embeddings, achieving an F1-score of 0.69 after 5000 training steps. Table 4 displays the F1-scores and accuracy of all three models after 5000 and 50000 training steps..

Model	F1 Score (5,000 steps)	F1 Score (50,000 steps)	Accuracy (5,000 steps)	Accuracy (50,000 steps)
CNN with word embeddings	0.70	0.69	0.93	0.93
CNN with character embeddings	0.57	0.73	0.92	0.94
LSTM with word embeddings	0.70	0.68	0.94	0.93

Table 5: Convolutional neural network results

Upon evaluating the CNN and LSTM models on Wikipedia comments, the researchers discovered that they were able to achieve accuracies of up to 0.94 in identifying comments in the test set that were not offensive or personal in nature. Interestingly, they found that character embedding yielded better results than lexical embedding for CNNs. While CNNs with character embeddings required more training iterations than CNNs and LSTMs with lexical embeddings, their computational performance was superior, despite having the same F1-score performance. Therefore, in terms of practical automated annotation management applications, the researchers recommend utilizing CNNs with character embeddings.

2.2 Analysis of the methods used

The objective of this study is to create two models, a basic Naive Bayes model and a deep learning recurrent neural network model with LSTM, which will be utilized on a common dataset with the aid of natural language processing (NLP) and machine learning algorithms. However, before delving into the workings of these models on the given dataset, it is crucial to comprehend the direct correlation between sentiment analysis and NLP and ML algorithms.

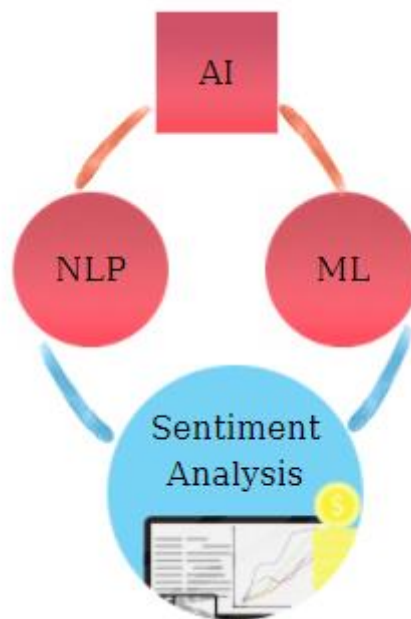


Figure 3 : *Relationship between NLP and ML*

Artificial Intelligence comprises various subcategories, including Machine Learning and Natural Language Processing, which are applied to tackle complex problems.

Machine Learning (ML): Machine learning, a branch of computer science, has evolved from the exploration of Computational Learning Theory and Pattern Recognition, which are part of Artificial Intelligence. The fundamental aim of machine learning is to create algorithms that can learn from available data and utilize this learning to make predictions. The ultimate goal is to

develop models that can extract decisions and make predictions as output, based on the data provided.

Natural Language Processing (NLP): Natural Language Processing is a subdivision of computer science and Artificial Intelligence that involves the interaction between human languages and computers. Its objective is to facilitate the production and comprehension of natural language, allowing computers to derive inferences and meanings from linguistic data.

Sentiment Analysis: The process of Sentiment Analysis involves analyzing users' opinions on various subjects such as products, social or political issues, etc. The insights obtained through Opinion Mining are valuable to several fields including customer service and marketing for businesses, scientists, and governments. These sectors utilize the results to promote their products, conduct research, and make informed decisions..

Opinion mining and Sentiment Analysis share a common purpose and can be seen as equivalent in some respects. However, while Opinion mining aims to recognize affirmative or negative expressions such as "I enjoy it" versus "I dislike it," Sentiment Analysis involves a broader examination of emotions and attitudes.

The procedure for extracting emotions can be outlined as follows:

1. **Using NLP Concepts of tokenization and ngraming:** A list of words is created based on a review or a text.
2. **Filtering the words (NLP concepts of Stop words processing):** Removing words that do not make sense in Opinion Mining (e.g. and, that, but, is).
3. **Root of words (NLP Concept of stemming and lemmatization):** Convert all words to their original root (e.g. cry,cries,cried -> cry).
4. **Detection of denial of words (Using NLP concepts of token processing):** Finding words that have the same meaning as the use of negation (e.g. not good = bad, not sad = happy etc.).
5. **Feature generations (Use feature extraction algorithms Bag of word, TF-IDF, Count vectorizer etc which - NLP again helps here):** Indication of Positivity or Negativity of a critique based on the use of words extracted as features from that critique.
6. **Create a Classification Model (Applying Machine Learning algorithms on features generated in previous steps to predict the polarity of the document):** Predicting Positive or Negative comments based on a Machine Learning model.

Three techniques are used in Sentiment Analysis:

1. **Machine learning:** Training and Development of a Classification model on already classified data with neutral, negative or positive content.
2. **Combined approach:** Using a method of an already classified algorithm and a dictionary
3. **Lexicon approach:** Locating text polarity by creating a dictionary.

To summarize, Machine Learning algorithms are taught using text, including comments, in order to develop the capability to classify other data based on its polarity (such as neutral, negative, or positive). Meanwhile, Natural Language Processing enables computers to comprehend text.

Module 3

3.1 Datasets

The data utilized in this study were previously used by Wassem and Hovy (2016) in English language Tweets on Twitter. These Tweets were divided into three separate json files. The first file (amateur_expert.json) contained 6,909 Tweets classified into four categories: Neither (5,850), Sexism (911), Racism (98), and Both (50). We removed the Both category for our experiment due to its small sample size, which could have led to incorrect conclusions. The second file contained 16,907 Tweets categorized as none (11,501), racism (1976), and sexism (3430). The third file was created from the second file and divided into two categories: abusive ("none") and none_abusive (sexism, racism) Tweets. This was done to produce results in two broader categories, allowing us to compare the accuracy and differences of the models on a different dataset. For the second file, we used data from the "Detecting Insults in Social Commentary" survey published by Impernium through Kaggle.

The comments in this work were labeled as '0' for neutral comments and '1' for corresponding insulting comments. It should be noted that this dataset has been used by other researchers, allowing us to compare our results with those of other models used for detecting abusive language on Twitter.

3.2 Naïve Bayes model

Naïve Bayesian models are a set of straightforward probabilistic classifiers in machine learning that apply Bayes' theorem with the assumption of strong independence between features. Since the 1950s, Naïve Bayes has been the subject of extensive research. In the early 1960s, it was introduced to the text retrieval community under a different name and has since remained a popular and basic method for text classification. This model utilizes word frequency to calculate the probability of a comment belonging to a specific category or another. The Naive Bayes model is easy to develop and particularly useful for very large datasets. Along with simplicity, Naive Bayes is known to outperform even very sophisticated classification methods.

But let's take a closer look at how the algorithm works.

Here is the Naïve Bayes equation:

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

Σχήμα 4: Naïve Bayes equation

According to the equation above,

- $P(A)$ is the probability that event A will occur
- $P(B)$ is the probability that event B will occur
- $P(A|B)$ is the probability that A occurs when B occurs
- $P(B|A)$ is the probability that B will occur when A occurs

In a broader sense, the Bayesian model is utilized to determine a probability when certain other probabilities are known.

However, to understand the model's application in practice, let's consider sentiment analysis. The table below displays movie reviews classified as negative or positive.

Id	Tweet	Class
1	<i>I loved the movie</i>	+
2	<i>I hated the movie</i>	-
3	<i>A great movie, good movie</i>	+
4	<i>Poor acting</i>	-
5	<i>a good movie, great acting</i>	+

Table 6: Movies Reviews

$P(+)=3/5=0,6$ -> three positive reviews

SUM of Unique_Words -> I, loved, the, movie, hated, a, great, good, poor, acting

N -> The words in the (+) case:14 words

N_k -> The number of times word k occurs in these cases (+)

$$P(W_k|+) = N_k+1/N+Unique_words$$

$$P(I|+) = 1+1/14+10 = 0,0833$$

$$P(the|+) = 1+1/14+10 = 0,833$$

$$P(a|+) = 2+1/14+10 = 0,125$$

$$P(acting|+) = 1+1/14+10 = 0,0833$$

$$P(hated|+) = 0+1/14+10 = 0,0417$$

$$P(loved|+) = 1+1/14+10 = 0,0833$$

$$P(movie|+) = 4+1/14+10 = 0,2083$$

$$P(great|+) = 2+1/14+10 = 0,125$$

$$P(good|+) = 2+1/14+10 = 0,125$$

$$P(poor|+) = 0+1/14+10 = 0,0417$$

$$P(-)=2/5=0,4$$

N -> The words in the (-) case:6

N_k -> The number of times word k occurs in these cases (+)

$$P(W_k|-) = N_k+1/N+Unique_words$$

$$P(I|-) = 1+1/6+10 = 0,125$$

$$P(the|-) = 1+1/6+10 = 0,125$$

$$P(a|-) = 0+1/6+10 = 0,0625$$

$$P(\text{acting}|-) = 1+1/6+10 = 0,125$$

$$P(\text{hated}|-) = 1+1/6+10 = 0,125$$

$$P(\text{loved}|-) = 0+1/6+10 = 0,06125$$

$$P(\text{movie}|-) = 1+1/6+10 = 0,125$$

$$P(\text{great}|-) = 0+1/6+10 = 0,0625$$

$$P(\text{good}|-) = 0+1/6+10 = 0,0625$$

$$P(\text{poor}|-) = 1+1/6+10 = 0,125$$

Now that we have trained our model let us classify a new review according to the following.

$$V_{nb} = \text{argmax} P(v_j) \prod P(W|V_j)$$

"I hated the poor acting"

$$\text{If } V_j = +; P(+)P(I|+)P(\text{hated}|+) P(\text{the}|+) P(\text{poor}|+) P(\text{acting}|+) = 6.03 \cdot 10^{-7}$$

$$\text{If } V_j = -; P(-)P(I|-)P(\text{hated}|-) P(\text{the}|-) P(\text{poor}|-) P(\text{acting}|-) = 1.22 \cdot 10^{-5}$$

Therefore the new review will belong to the negative comments.

3.2.1 Naïve Bayes train/test set

During our experiment, we trained the Bayesian model using the `amateur_expert.json` dataset, which provided detailed information about each tweet, including the tag of each comment (e.g., Abusive or non-Abusive), the date the tweet was written, the user ID, and the number of followers. However, to use this data for training, we needed to extract only the comment tags. Since the original file was in JSON format, which is not easy to edit, we converted it to a Python format and created a new file called `data.txt` to make it easier for the Bayesian model to manage. This new file contained only the comment tags and their corresponding tweets separated by commas. To extract only the comment tags, we used a useful Python tool called a data frame, which allows for easy access and visualization of two-dimensional tables with columns of different data. We utilized data frames to categorize tweets as comments, and separate them into respective categories, allowing us to manipulate the table data with ease and create dictionaries for each category. We developed three separate dictionaries for the categories of "neither," "sexism," and "racism," which were then processed to remove extraneous words such as punctuation marks, articles, prepositions, and other insignificant words. We then split the data into a training set and a testing set, with 80% going to the former and 20% going to the latter. To train the algorithm, we utilized `NaiveBayesClassifier` from the `ntlk.classify` library, and to evaluate

the model's accuracy, we used the test set we had developed. Finally, we tested the model on additional tweets that were not present in our existing data to determine if it could provide valid results.

The Naïve Bayes experiment yielded an accuracy of 33%, which is considered a low score. Therefore, it's crucial to examine how this score was attained and identify methods to enhance it.

The illustrations below showcase the tweets for each category (Neither, Racism, and Sexism), the model's accuracy, and the outcomes obtained from feeding the model with various comments.

```
C:\Python\Python36-32>python abusive_Naive1.py
The Neither category contains: 5844 tweets
The Racism category contains: 98 tweets
The Sexism category contains: 909 tweets
The Accuracy is: 32.96783625730994
The tweet "Women should not drive!" belongs to : Sexism category
The tweet "I forgot my phone" belongs to : Neither category
The tweet "Israel is S000 #racist - sure. after the #arab #terror #attack today" belongs to : Racism category
C:\Python\Python36-32>_
```

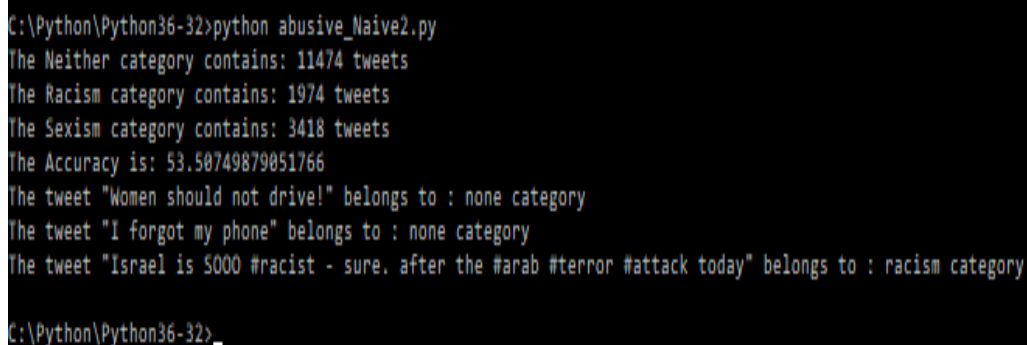
Figure 5: abusive_Naive1.py results

As we can see in the snapshot above, the low accuracy of the model is first of all due to the fact that the tweets in the neither category are more than in the racism or sexism category. Also, the tweets in the racism category may also be in the sexism category as these two categories may be related to each other. This practically means that some words that are in the dictionary in the racism category may also be in the sexism category, so that the algorithm is unable to give the correct categorization of the tweet.

3.2.2 Application of Naïve Bayes on new data

As we mentioned the accuracy of Naïve Bayes in the `amateur_expert.json` file was quite low, for this purpose the model was applied to the same dataset except that this time we used more tweets in the racism and sexism categories. The file this time contained 11,473 tweets in the neither category, 1,974 tweets in the racism category and 3,418 tweets in the sexism category. So we noticed that even if the tweets in the neither category are still more the accuracy achieved was 53.5%.

The following screenshot shows the results from the second experiment using a different set of tweets from the same dataset.



```
C:\Python\Python36-32>python abusive_Naive2.py
The Neither category contains: 11474 tweets
The Racism category contains: 1974 tweets
The Sexism category contains: 3418 tweets
The Accuracy is: 53.50749879051766
The tweet "Women should not drive!" belongs to : none category
The tweet "I forgot my phone" belongs to : none category
The tweet "Israel is 5000 #racist - sure. after the #arab #terror #attack today" belongs to : racism category
C:\Python\Python36-32>
```

Figure 6: abusive_Naive2.py results

However even if the accuracy was higher the prediction according to the above tweet "Women should not drive!" belongs to the sexism category and not to none as noted in the above image.

The third experiment we tried to achieve was to apply our model to only two categories to see if we could achieve a better score. For this purpose we divided the tweets into only two categories (abusive, noneAbusive [racism,sexism]). The accuracy of the model reached about 56% which is the best result we obtained using the Naïve Bayes model.

Here is the result of the experiment.

```
C:\Python\Python36-32>python abusive_noneAbusiveNaive.py
The None Abusive category contains: 11472 tweets
The Abusive category contains: 5392 tweets
The Accuracy is: 56.31672597864769
The tweet "Women should not drive!" belongs to : abusive category
The tweet "I forgot my phone" belongs to : none_abusive category
The tweet "he is so racist" belongs to : abusive category

C:\Python\Python36-32>
```

Figure 7: abusive – none abusive results

The last experiment we tried was to apply the algorithm to a different data set. The data was extracted from Impernium via Kaggle (Detecting Insults in Social Commentary) and contains two tags insult (1) or neutral (0). The reason why we trained and tested the algorithm on new data is to see the accuracy and flexibility of the model on data where the algorithm has not been trained. The accuracy we got from the last experiment was at 78%. The figure below shows our results from the last experiment and the dataset we used in each category.

```
C:\Python\Python36-32>python insults_Naive.py
The Neutral category contains: 2898 tweets
The Insult category contains: 1049 tweets
The Accuracy is: 78.42639593908629
The tweet "fuck off" belongs to : 1 category
The tweet "people should not talk to you cause you are a retard!" belongs to : 1 category
The tweet "you are so polite and generous these dayss" belongs to : 0 category

C:\Python\Python36-32>
```

Figure 8: insults_Naive.py

The experiment yielded a considerably high score, and the accuracy of the tweets' results was confirmed.

The Naïve Bayes algorithm was employed for all the experiments, and the outcomes are listed in the table below..

Class	Experiment1	Experiment2
Neither/none	5844	11474
Sexism	909	3418
Racism	98	1974
Score	33%	54%

Table 7 : *Results from the first two experiments*

Class	Experiment3
None_Abusive	11472
Abusive	5392
Score	56%

Table 8: Results from the third experiment

Class	Experiment4
Neutral	2898
Insults	1049
Score	78%

Table 9: Results from the fourth experiment, applied to a new data set

3.3 Recurrent Neural Networks

Recurrent neural networks belong to a category of artificial neural networks that mimic the topology and neuron structure of the human brain. Their objective is to learn from sequential data in the order it is presented, and generate new content based on it. Unlike traditional neural networks, where inputs and outputs are assumed to be independent of each other, RNN models aim to develop information in a sequence. For instance, to predict the next word in a sentence, it is beneficial to have knowledge of the previous word.

These models are referred to as recurrent, as they perform the same function for each element in a sequence, and their output depends on prior computations. An essential advantage of RNN models is their exceptional memory, as they can store information for a given period. Although RNNs can theoretically use information from sequences of any length, in practice, they can only consider a few steps back.

The following figure illustrates the sequence of an RNN model.

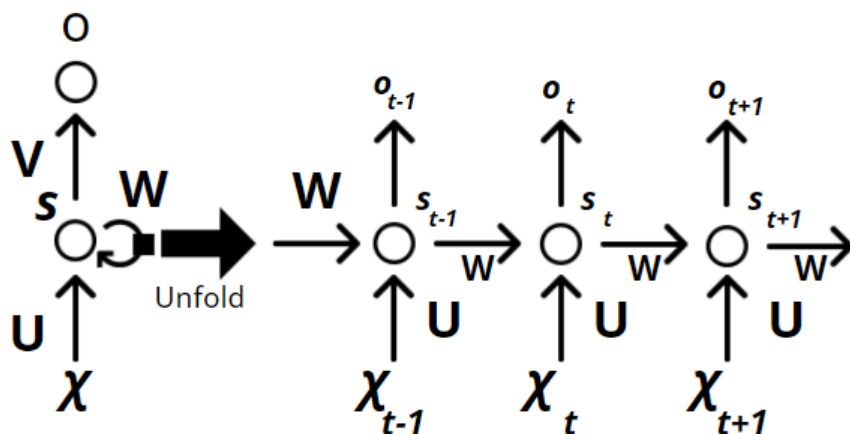


Figure 9: A recurrent neural network and its evolution in computation time involved in its future computation.

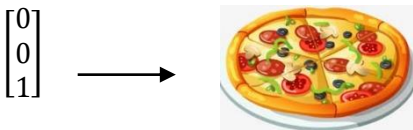
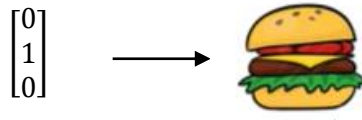
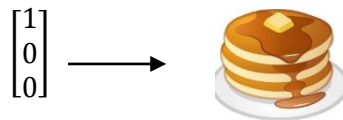
Η παραπάνω εικόνα μας δείχνει πώς το μοντέλο RNN ξεδιπλώνεται (unfolding) σε ένα ολοκληρωμένο δίκτυο. Με τον όρο ότι το μοντέλο ξεδιπλώνεται εννοούμε ότι μπορούμε να δούμε αναλυτικά κάθε τμήμα της ακολουθίας του δικτύου. Για παράδειγμα, ας υποθέσουμε ότι έχουμε μια πρόταση 6 λέξεων, το δίκτυο θα ξεδιπλωθεί σε 6 στρώματα, ένα στρώμα για κάθε λέξη.

The depicted figure exhibits the unfolded version of an RNN model integrated into a network, which allows us to inspect each segment of the network sequence in detail. For instance, if we consider a sentence comprising six words, the network will unfold into six layers, with one layer for each word.

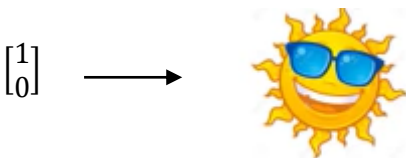
To enhance our comprehension of the neural network, we devised a straightforward example inspired by food and weather. Let's assume that we have the world's best chef who prepares diverse dishes daily, contingent upon the weather. Suppose we have three distinct dishes and a vector representing each one.

Example of a neural network:

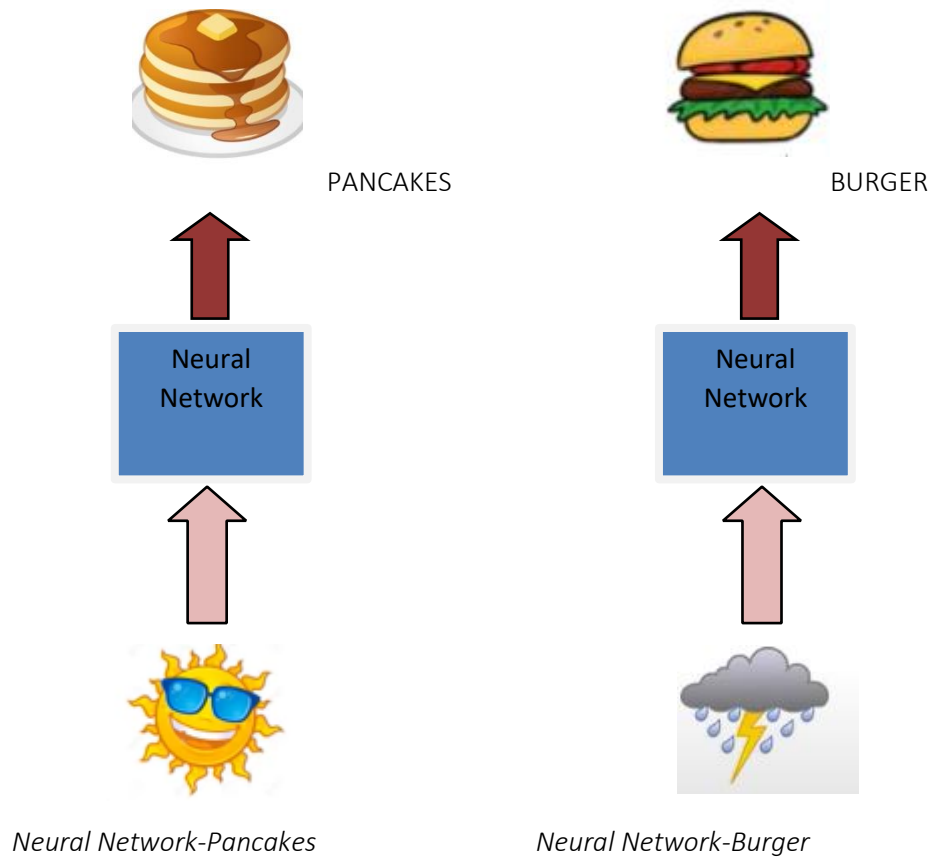
Food vectors (Pancakes, Burger, Pizza)



Διανύσματα καιρού (sunny, rainy)



We can presume that the chef prepares pancakes when the weather is sunny, and burgers when it's rainy.



So if we multiply the two tables Pancakes, Burger with Sunny and Rainy respectively the results we will get will be the following:

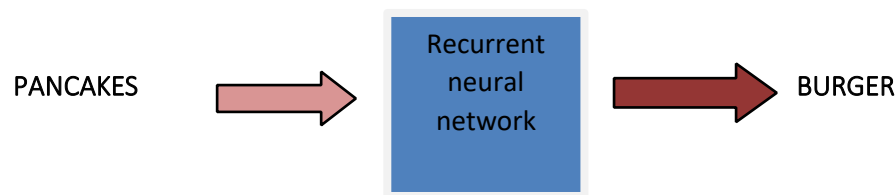
if the weather is sunny

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \text{Pancakes}$$

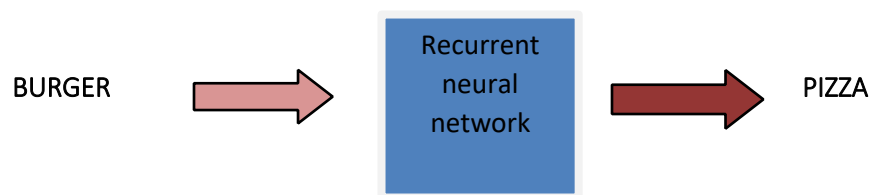
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \text{Burger}$$

Now let's see how a recurrent neural network (RNN) works.

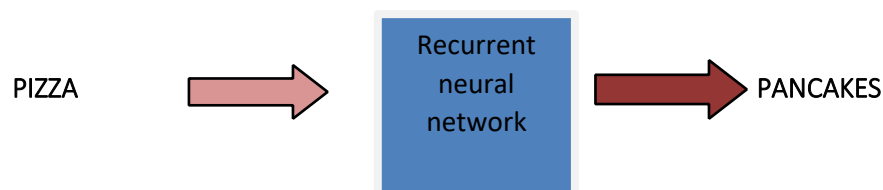
In this instance, the chef follows a sequential cooking pattern. For instance, on Monday, he prepares pancakes, on Tuesday, he makes burgers, and on Wednesday, he creates pizza. Then on Thursday, he starts again with pancakes, followed by burgers, and pizza the next day.



Recurrent Neural Network- Burger



Recurrent Neural Network- Pizza



Recurrent Neural Network- Pancakes

Let's see the results based on the three dishes the Chef cooks during the week.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \longrightarrow \text{Burger}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \longrightarrow \text{Pizza}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \longrightarrow \text{Pancake}$$

As demonstrated in the RNN model, it employs the output as input each time, which designates it as a recurrent neural network. This mechanism enables the model to develop memory and perceive its next step.

3.3.1 RNN train/test set

To begin with, the RNN model underwent training on the same dataset as Naïve Bayes (amateur_expert.json). Initially, we utilized the data.txt file and then created a dataframe to split the data into two columns for ease of access and better management (category, tweets), as in our previous algorithm. Concerning data pre-processing, we implemented the lambda function to convert capitalized words to lower case and eliminate punctuation marks, which do not offer any value to our text or algorithm. In general, the RNN model employs the Keras library, which includes practical functions for data pre-processing. Hence, to separate words in the tweet comments, we employed the tokenizer function from the Keras library. Moreover, it's worth mentioning that the RNN model's predictions use the most frequent words in the text, which makes it particularly intelligent compared to the Naïve Bayes model. In this regard, we utilized 2000 words for training the algorithm, where each word comprises random integers between 0-2000, as machine learning algorithms employ integers instead of words for training. Our developed model is a Keras sequence model, which constitutes a linear stack of layers. The initial layer is the Embedding Layer, which converts input data into fixed-size vectors. The second layer is the LSTM layer for short-term memory, which can endure for an extended duration. The LSTM layer is capable of capturing time series relationships with unknown delays between important events. This characteristic of being relatively insensitive to the temporal distance between information is the main advantage of Long Short-Term Memory Networks over Markovian models and other learning techniques. Next, we added a Dense layer which applies the softmax function to generate the correct output for each vector. Similar to the Naïve Bayes model, tweets were categorized into neither, racism, and sexism, with corresponding vectors of [1 0 0], [0 1 0], and [0 0 1], respectively. The data was then divided into training and testing sets using the train_test_split function from the sklearn library. The model was then trained using the training data and evaluated for accuracy on the testing data. In the first experiment, the model was trained for 15 epochs using 32 annotations (tweets) and achieved an accuracy of approximately 86%, which is a remarkable performance for the RNN model. The accuracy of the model and the prediction of a tweet belonging to the sexism category based on the chosen vector are shown in the following figure.

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 104, 128)	2560000
spatial_dropout1d_1 (Spatial	(None, 104, 128)	0
lstm_1 (LSTM)	(None, 196)	254800
dense_1 (Dense)	(None, 3)	591

=====
 Total params: 2,815,391
 Trainable params: 2,815,391
 Non-trainable params: 0
 =====

```

None
[1 0 0]
[0 1 0]
(4590, 104) (4590, 3)
(2261, 104) (2261, 3)
Epoch 1/15
- 59s - loss: 0.4664 - acc: 0.8466
Epoch 2/15
- 55s - loss: 0.2629 - acc: 0.9026
Epoch 3/15
- 56s - loss: 0.1620 - acc: 0.9414
Epoch 4/15
- 57s - loss: 0.0929 - acc: 0.9712
Epoch 5/15
- 57s - loss: 0.0567 - acc: 0.9832
Epoch 6/15
- 57s - loss: 0.0361 - acc: 0.9911
Epoch 7/15
- 55s - loss: 0.0292 - acc: 0.9913
Epoch 8/15
- 56s - loss: 0.0176 - acc: 0.9946
Epoch 9/15
- 58s - loss: 0.0165 - acc: 0.9961
Epoch 10/15
- 58s - loss: 0.0147 - acc: 0.9952
Epoch 11/15
- 57s - loss: 0.0124 - acc: 0.9963
Epoch 12/15
- 58s - loss: 0.0109 - acc: 0.9972
Epoch 13/15
- 58s - loss: 0.0082 - acc: 0.9972
Epoch 14/15
- 58s - loss: 0.0144 - acc: 0.9967
Epoch 15/15
- 57s - loss: 0.0074 - acc: 0.9980
score: 0.71
acc: 86.85939546170277
[[0.35815695 0.0393987 0.60244435]]
[[0 0 1]]
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
Sexism

```

Figure 10: Recurrent Neural Network-RNN.py

3.3.2 Application of RNN model on new data

In our second experiment, we utilized the same two categories as in the Naïve Bayes model (none_abusive, abusive). The dataset was obtained from the data2.txt file and consisted of 11,472 none-abusive tweets and 5,392 abusive tweets. However, this time we made a slight modification to the dense layer by reducing the number of categories from 3 to 2. This resulted in the vector representation [0 1] for the none_abusive category and [1 0] for the abusive category. The model was trained for 15 epochs using 32 tweets and then for 5 epochs using 50 tweets. As can be seen in the figure below, the accuracy of the model was 78% and 80% for the two experiments, respectively.

```
Using TensorFlow backend.
The None Abusive category contains: 11474 tweets
The Abusive category contains: 5392 tweets
```

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 104, 128)	2560000
spatial_dropout1d_1 (Spatial	(None, 104, 128)	0
lstm_1 (LSTM)	(None, 196)	254800
dense_1 (Dense)	(None, 2)	394

```
=====  
Total params: 2,815,194  
Trainable params: 2,815,194  
Non-trainable params: 0  
=====  
None  
[0 1]  
[1 0]  
(11300, 104) (11300, 2)  
(5566, 104) (5566, 2)  
Epoch 1/5  
- 135s - loss: 0.5203 - acc: 0.7559  
Epoch 2/5  
- 130s - loss: 0.3298 - acc: 0.8625  
Epoch 3/5  
- 132s - loss: 0.2215 - acc: 0.9136  
Epoch 4/5  
- 143s - loss: 0.1501 - acc: 0.9440  
Epoch 5/5  
- 135s - loss: 0.1105 - acc: 0.9588  
score: 0.63  
The accuracy is: 79.78357030321564  
[[0.9862742 0.01372587]]  
[[1 0]]  
The tweet ['all psychopaths have a female brain'] is Abusive
```

Figure 11: Recurrent Neural Network-RNN.py

```

The None Abusive category contains: 11474 tweets
The Abusive category contains: 5392 tweets

Layer (type)                 Output Shape              Param #
=====
embedding_1 (Embedding)      (None, 104, 128)         2560000
spatial_dropout1d_1 (Spatial (None, 104, 128)         0
lstm_1 (LSTM)                 (None, 196)              254800
dense_1 (Dense)              (None, 2)                 394
=====
Total params: 2,815,194
Trainable params: 2,815,194
Non-trainable params: 0

None
[0 1]
[1 0]
(11300, 104) (11300, 2)
(5566, 104) (5566, 2)
Epoch 1/10
- 141s - loss: 0.5238 - acc: 0.7606
Epoch 2/10
- 138s - loss: 0.3229 - acc: 0.8642
Epoch 3/10
- 138s - loss: 0.2195 - acc: 0.9140
Epoch 4/10
- 139s - loss: 0.1483 - acc: 0.9447
Epoch 5/10
- 136s - loss: 0.1052 - acc: 0.9616
Epoch 6/10
- 136s - loss: 0.0817 - acc: 0.9692
Epoch 7/10
- 552s - loss: 0.0658 - acc: 0.9770
Epoch 8/10
- 138s - loss: 0.0544 - acc: 0.9804
Epoch 9/10
- 142s - loss: 0.0437 - acc: 0.9839
Epoch 10/10
- 137s - loss: 0.0366 - acc: 0.9867
score: 0.85
The accuracy is: 78.40629611411707
[[9.9951327e-01 4.8676669e-04]]
[[1 0]]
The tweet ['all psychopaths have a female brain'] is Abusive

```

Figure 12: Recurrent Neural Network-RNN.py

It should be noted that there exist various techniques to enhance the accuracy of the model. To that end, we attempted to decrease the number of training epochs (from 5 to 3) as well as the number of tweets used for training (from 50 to 32). As demonstrated in the subsequent figure, the model's accuracy increased to approximately 82%.

```

The None Abusive category contains: 11474 tweets
The Abusive category contains: 5392 tweets

```

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 104, 128)	6400000
spatial_dropout1d_1 (Spatial)	(None, 104, 128)	0
lstm_1 (LSTM)	(None, 196)	254800
dense_1 (Dense)	(None, 2)	394

```

Total params: 6,655,194
Trainable params: 6,655,194
Non-trainable params: 0

```

```

None
[0 1]
[1 0]
(11300, 104) (11300, 2)
(5566, 104) (5566, 2)
Epoch 1/3
- 179s - loss: 0.5064 - acc: 0.7669
Epoch 2/3
- 175s - loss: 0.3133 - acc: 0.8740
Epoch 3/3
- 176s - loss: 0.1976 - acc: 0.9246
score: 0.46
The accuracy is: 81.52975897688147
[[0.97422534 0.02577473]]
[[1 0]]
The tweet ['all psychopaths have a female brain'] is Abusive

```

Figure 13: Recurrent Neural Network-RNN.py

After conducting several experiments on the initial dataset, it was observed that the RNN model could not surpass an accuracy of 86%. Therefore, a new dataset was used, which was also used in the Naïve Bayes model, to assess the model's performance on comments where it has not been trained and to compare it with the Naïve Bayes model. This dataset, called Impernium via Kaggle, contained only two labels, insult class(1) and neutral class(0). The dataset consisted of 1,049 tweets in the insult category and 2,898 in the neutral category. The model was trained for 10 epochs with a batch size of 32 tweets, and the accuracy obtained was around 80%, as shown in the figure below. This is a satisfactory result, considering that the model was trained on a different dataset than the one used in previous experiments.

```

Insult      Date      Comment
0      1  20120618192155Z  "You fuck your dad."\t\t\t\t\t\t
1      0  20120528192215Z  "i really don't understand your point.\xa0 It ...
2      0      NaN  "A\\xc2\\xa0majority of Canadians can and has ...
3      0      NaN  "listen if you dont wanna get married to a man...
4      0  20120619094753Z  "C\\xe1c b\\u1ea1n xu\\u1eding \\u0111\\u01b0\\u1edd...
The Neutral category contains: 2898 tweets
The Insult category contains: 1049 tweets
[ 0  0  0 ... 70 14 691]

Layer (type)              Output Shape              Param #
=====
embedding_1 (Embedding)    (None, 2394, 128)         2560000
spatial_dropout1d_1 (Spatial (None, 2394, 128)         0
lstm_1 (LSTM)              (None, 196)               254800
dense_1 (Dense)            (None, 2)                 394
=====
Total params: 2,815,194
Trainable params: 2,815,194
Non-trainable params: 0

None
[0 1]
[1 0]
(2644, 2394) (2644, 2)
(1303, 2394) (1303, 2)
Epoch 1/10
- 4647s - loss: 0.5541 - acc: 0.7383
Epoch 2/10
- 1075s - loss: 0.3707 - acc: 0.8347
Epoch 3/10
- 975s - loss: 0.2182 - acc: 0.9198
Epoch 4/10
- 967s - loss: 0.1170 - acc: 0.9565
Epoch 5/10
- 981s - loss: 0.0886 - acc: 0.9701
Epoch 6/10
- 980s - loss: 0.0471 - acc: 0.9837
Epoch 7/10
- 957s - loss: 0.0519 - acc: 0.9856
Epoch 8/10
- 945s - loss: 0.0124 - acc: 0.9974
Epoch 9/10
- 1212s - loss: 0.0176 - acc: 0.9951
Epoch 10/10
- 1064s - loss: 0.0139 - acc: 0.9958
score: 0.94
The accuracy is: 80.12279350301988

```

Figure 14: Recurrent Neural Network-RNN.py

```

    Insult      Date      Comment
0      1  20120618192155Z  "You fuck your dad."\t\t\t\t\t\t\t
1      0  20120528192215Z  "i really don't understand your point.\xa0 It ...
2      0      NaN  "A\\xc2\\xa0majority of Canadians can and has ...
3      0      NaN  "listen if you dont wanna get married to a man...
4      0  20120619094753Z  "C\\xe1c b\\ulea1n xu\\uleding \\u0111\\u01b0\\uledd...
The Neutral category contains: 2898 tweets
The Insult category contains: 1049 tweets
[ 0  0  0 ... 70 14 691]

Layer (type)                Output Shape                Param #
=====
embedding_1 (Embedding)      (None, 2394, 128)          2560000
spatial_dropout1d_1 (Spatial (None, 2394, 128)          0
lstm_1 (LSTM)                (None, 196)                254800
dense_1 (Dense)              (None, 2)                   394
=====
Total params: 2,815,194
Trainable params: 2,815,194
Non-trainable params: 0

None
[0 1]
[1 0]
(2644, 2394) (2644, 2)
(1303, 2394) (1303, 2)
Epoch 1/3
- 987s - loss: 0.5508 - acc: 0.7405
Epoch 2/3
- 999s - loss: 0.3780 - acc: 0.8404
Epoch 3/3
- 1798s - loss: 0.2429 - acc: 0.8986
score: 0.44
The accuracy is: 81.81120487972122
[[0.91994965 0.08005028]]
[[1 0]]
The comment ['all of them are bunch of retards!!'] is Insult

```

Figure 15: Recurrent Neural Network-RNN.py

The previous figure demonstrates that we can achieve a slightly improved result by reducing the epochs by 1%. The tables below summarize all the experiments conducted using the RNN algorithm.

Class	Experiment1
Neither	5844
Sexism	909

Racism	98
Score	87%

Table 10: RNN_1.py

Class	Experiment2
None_Abusive	11472
Abusive	5392
Score	78%

Table 11: RNN_2.py

Class	Experiment3
Neutral	2898
Insults	1049
Score	80%

Table 12: RNN_3.py

Module 4

4.1 Discussion of results

As presented in this thesis, two algorithms were developed to identify abusive language in social media. The first one was Naïve Bayes, which uses natural language processing (NLP) as a simple probability model. The second one was the recurrent neural network (RNN), which is a machine

learning model that employs the Keras library. Both models were tested using the same dataset in order to allow for a comparison of their results. It should be noted that this dataset has been used by other researchers in the past, making its selection non-random. For example, Wassem and Hovy (2016) trained their model on this dataset by evaluating different characteristics such as gender and location, using a logistic regression classifier. The best result they achieved was 77.74%. Similarly, Bourgonje, Moreno-Scheider, and Ankit Srivastan (2018) used this dataset and achieved the highest score of 80% among three different datasets by employing a specific combination of features such as character bi-to-fourgrams and gender. Gamback and Sikdar (2017) also utilized this dataset and applied a CNN model to obtain a result of 78.29%. As seen from these experiments, all models produced results with only small differences, except for Bourgonje et al. (2018) who developed a specific combination of features to achieve the highest score. In this research, two models were developed, a simple probability model and a machine learning model, and it was concluded that the RNN model provided the best results. Initially, Naïve Bayes was applied to the dataset used by Wassem and Hovy (2016). The dataset was initially split into three classes: "neither," "racism," and "sexism." Unfortunately, the achieved score was quite low, coming in at almost 33%. This was likely due to the `amateur_expert.json` file containing many more tweets in the "neither" class compared to the other two. In the second experiment, we attempted to split the tweets into more proportional groups between the classes, resulting in a better score of 54%. We then tried to split the tweets into only two categories: "abusive" and "none_abusive" (combining "racism" and "sexism"), and this resulted in a 2% increase in accuracy. However, since the dataset was still providing a relatively low score, we attempted to train the model on a different dataset that only contained two classes ("Detecting Insults in Social Commentary"). The resulting score was much better, reaching 78%. Therefore, based on our experiments, we concluded that the Naïve Bayes model, being a probabilistic model, is highly reliant on the percentage of each class that the model is trained on, the type of comments used, and how NLP manages our data. We will provide more detail on how to manage NLP and use it to achieve better results in the next section.

The RNN model, which was the second algorithm we developed, was applied to the same dataset, and we were able to achieve an accuracy of 87%. It's worth noting that the machine learning model produced an excellent result even when the data was not appropriately distributed across the categories mentioned above. We trained the model using 15 epochs on 32 Twitter comments for this experiment. In the second experiment, we attempted, like in the Naïve Bayes model, to split the data into abusive (racism, sexism) and non-abusive tweets categories, achieving a score of 82% using 3 epochs and 32 tweets. Then, for these tweets, we conducted additional experiments with varying epochs and batch sizes (number of comments) to compare the results. The detailed outcomes are presented in section (3.3.2 Applying the RNN model to a new dataset).

It was observed that the Naïve Bayes model provided superior results when the proportions between categories were distributed properly. However, when applied to data belonging to the categories of neither, sexism, and racism, the model faced difficulties due to its bag of words approach during training. This resulted in a significant reduction in accuracy as a word in one category could also appear in another. Additionally, the model was limited to predicting words it had seen before during its training.

On the other hand, the RNN model did not require a proper comment ratio to achieve good results, as different ratios resulted in fairly high accuracies by adjusting the model parameters. The RNN model was observed to outperform the Naïve Bayes model in terms of accuracy.

Lastly, it is worth noting that both models were quite flexible, as the code was well-structured and could work on any dataset with different results achievable by changing the data.

4.2 Suggestions for future work

This paper presents an analysis of the two algorithms and their respective results, achieved through various techniques applied in our experiments. Although the results obtained were high, there is room for further improvement of both models through several techniques. For instance, in the case of Naïve Bayes, we utilized NLP techniques such as the removal of stopwords (i.e., punctuation marks), but there are other words in the text that may not provide any value to our comments and could also be removed. The category of these is called:

language stopwords και περιλαμβάνει τις ακόλουθες υποκατηγορίες:

Location Stopwords – Country names, Cities names etc

Time Stopwords – Name of the months and days (January, February, Monday, Tuesday, today, tomorrow, etc)

Numerals Stopwords – Words describing numerical terms (hundred, thousand, etc)

Furthermore, eliminating words that occur with very low frequency in our comment set can also improve our results. Another crucial approach that could greatly enhance the accuracy of our models is lemmatization, which involves transforming all words to their original root form.

For instance, the words "playing," "player," "plays," and "play" would all be transformed to "play." Additionally, in some cases, combining words can yield better comprehension of the text than individual words alone.

This technique, known as N-grams, is most commonly applied through the use of Bigrams. The following table illustrates the implementation of the Bigrams technique.

Old Features	Modified Features
love	love phone
phone	ate apple
apple	apple + NOUN
book	book + VERB

It is important to mention that machine learning models have a different approach in terms of their improvement techniques. For example, the RNN model has much higher accuracy compared to the Naïve Bayes model, however with some further methods we could improve our model results considerably. In general these models work much better using much more data so assuming we cannot change our data and add new data we can simply create more random vectors from the existing data since the model as mentioned in a previous section works with 0.1 vectors. Finally, we could make different parameterizations to the model by using more epochs and layers as well as larger batch -size (annotations). All these further techniques for both the Naïve Bayes and RNN model are very important steps for the next steps at the research level in machine learning algorithms.

Module 5

Bibliography

1. Anon., 2018. *ALW2: 2nd Workshop on Abusive Language Online*. [Online]
Available at: <https://sites.google.com/view/alw2018/call-for-papers>
[Accessed 1 August 2021].
2. Bourgonje, P., Moreno-Schneider, J., Srivastava, A. & Rehm, G., 2018. [Online]
Available at: https://link.springer.com/chapter/10.1007/978-3-319-73706-5_15
[Accessed 1 August 2021].
3. Gamback, B. & Sikdar, U. K., 2017. [Online]
Available at:
<https://pdfs.semanticscholar.org/0dca/29b6a5ea2fe2b6373aba9fe0ab829c06fd78.pdf>
[Accessed 1 August 2021].

4. Ho Park, J. & Fung, P., n.d. [Online]
Available at: <https://arxiv.org/abs/1706.01206>
[Accessed 1 August 2021].
5. Processing, N. L., n.d. *Wikipedia*. [Online]
Available at: https://en.wikipedia.org/wiki/Natural_language_processing
[Accessed 1 August 2021].
6. Waseem, Z. & Hovy, D., 2016. [Online]
Available at: <http://www.aclweb.org/anthology/N16-2013>
[Accessed 1 August 2021].
7. Wikipedia, n.d. *Naive Bayes Classifier*. [Online]
Available at: https://en.wikipedia.org/wiki/Naive_Bayes_classifier
[Accessed 1 August 2021].
8. Wikipedia, n.d. *Recurrent Neural Network*. [Online]
Available at: https://en.wikipedia.org/wiki/Recurrent_neural_network
[Accessed 1 August 2021].
9. Wikipedia, n.d. *Sentiment Analysis*. [Online]
Available at: https://en.wikipedia.org/wiki/Sentiment_analysis
[Accessed 1 August 2021].