Netspar

Network for Studies on Pensions, Aging and Retirement

NETSPAR ACADEMIC SERIES

The Impact of Stemming and Lemmatization Applied to Word Vector Based Models in Sentiment Analysis

A Comparison of Stemming and Lemmatization Methods

Youri Senders

MSc 06/2021-010

THE IMPACT OF STEMMING AND LEMMATIZATION APPLIED TO WORD VECTOR BASED MODELS IN SENTIMENT ANALYSIS

A COMPARISON OF STEMMING AND LEMMATIZATION METHODS

YOURI SENDERS

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
AT THE SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
OF TILBURG UNIVERSITY

STUDENT NUMBER

2018966

COMMITTEE

prof. dr. E.O. Postma
C.D. Emmery MSc

LOCATION

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science &
Artificial Intelligence
Tilburg, The Netherlands

DATE

June 25, 2021

ACKNOWLEDGMENTS

Many thanks to Tilburg University and its professors for teaching the courses I took during the Master's program. With special thanks to prof. dr. E.O. Postma for supervising me during my thesis.

THE IMPACT OF STEMMING AND LEMMATIZATION APPLIED TO WORD VECTOR BASED MODELS IN SENTIMENT ANALYSIS

A COMPARISON OF STEMMING AND LEMMATIZATION METHODS

YOURI SENDERS

Abstract

Sentiment analysis is a commercially attractive field in which a great deal of research has already been conducted. Millions of people across the world share their opinion on the web, which makes it a popular subject among researchers. The first step in sentiment analysis is text preprocessing. This research focuses on the methods stemming and lemmatization. These are text normalization methods that attempt to obtain root forms of inflected words. Prior work highlighted the importance of preprocessing. However, while preprocessing in classification models is studied extensively, little work has been done towards preprocessing in word vector based models. Therefore, the goal of this work is to examine the role of stemming and lemmatization when applied at the training phase of word vector based models. The following research question is addressed: "Which stemming or lemmatization method is most suitable for predicting sentiment polarity when integrated at the training phase of word vector based models?" This thesis uses a training corpus consisting of 142.570 news articles and the IMDB movie review dataset for classification. First, stemming and lemmatization are applied to the training corpus. Second, Word2Vec's CBOW and Skip-gram models are trained. Hereafter, stemming and lemmatization are applied to the classification dataset to obtain a compatible vocabulary. Finally, sentiment is classified using a LSTM model with embedding layer. The results of the experiments show that all methods outperformed the baseline for both word embedding models. Lemmatization is the preferred method for the CBOW model, whereas the Snowball stemmer is the preferred stemming method. The Snowball stemmer achieved the best performance for the Skip-gram model, while the Porter stemmer and lemmatization are close behind. Therefore, this research concludes that the Snowball stemmer is the best performing stemming method, while lemmatization achieves similar results.

1 INTRODUCTION

Affective tasks in natural language processing such as sentiment analysis are a popular research field in which a great deal of research has already been conducted (Babanejad, Agrawal, An, & Papagelis, 2020). In sentiment analysis, an opinion from a writer towards a specific event, personality or product is identified and categorized (Alam & Yao, 2019). Identifying sentiment can be beneficial for consumers before making a purchase, but also the other way around: organizations that want to know how consumers think about them (Go, Bhayani, & Huang, 2009). Traditional sentiment analysis started more than a decade ago by focusing on large amounts of text such as movie reviews. The rise of social media provided the opportunity for users to express their view on topics (Jianqiang & Xiaolin, 2017; Krouska, Troussas, & Virvou, 2016; Symeonidis, Effrosynidis, & Arampatzis, 2018).

Predicting the polarity of a piece of text in order to find out whether the writer expresses a positive, neutral or negative opinion is a common task in sentiment analysis (Alam & Yao, 2019; Angiani et al., 2016). Machine learning classification algorithms are mostly involved to solve a task like this. A machine learning classifier aims to associate a given input with the correct class. To illustrate, if a product review is given to the classifier, it could predict that the review has a positive sentiment. An algorithm needs to be trained on a set of inputs and their corresponding classes. Then, the classifier becomes capable of recognizing the correct class by extracting features from input data (Angiani et al., 2016).

The quality of the data is a major factor for machine learning classification algorithms. Inadequate data could result in a classifier that produces less accurate results (Kotsiantis, Kanellopoulos, & Pintelas, 2006). A corpus of raw text data can be unstructured. Thus, the first step in sentiment analysis is preprocessing the data (Angiani et al., 2016; Camacho-Collados & Pilehvar, 2017; Jianqiang & Xiaolin, 2017; Symeonidis et al., 2018). Within this phase, a series of techniques are applied to make the data more uniform and structured in preparation for training a machine learning algorithm. Preprocessing is considered as a crucial phase in order to achieve valuable results (Angiani et al., 2016; Babanejad et al., 2020; Kotsiantis et al., 2006; Krouska et al., 2016; Symeonidis et al., 2018). However, the disposal of noisy instances is a challenging task in machine learning (Kotsiantis et al., 2006).

Stemming and lemmatization are text normalization techniques that can be used to handle mismatches in language and are widely used in natural language processing tasks such as sentiment analysis, web search results, and information retrieval. Stemming is a technique that equate several variants of word forms to the root word called a "Stem". A stemmer uses rule-based heuristics in order to remove prefixes and suffixes to create matching Stems.

For example, the words "automatic", "automate", and "automation" all get the Stem "automat" (Asghar, Khan, Ahmad, & Kundi, 2014). It is possible that a Stem is a non-valid word in language (Asghar et al., 2014; Hickman, Thapa, Tay, Cao, & Srinivasan, 2020; Jivani et al., 2011).

Lemmatization is a similar technique. The main difference is that lemmatization takes the part of speech of words into account and returns a root word, called a "Lemma", by indexing through a dictionary. For instance, the words "plays", "played", and "playing" all get the Lemma "play" (Asghar et al., 2014). As a result, the lemmatization technique returns a valid word in language. Hence, lemmatization replaces all the words with their root form, whereas stemming only removes word inflections (Asghar et al., 2014; Hickman et al., 2020; Mhatre, Phondekar, Kadam, Chawathe, & Ghag, 2017).

According to Asghar et al. (2014), lemmatization is more accurate but computationally more expensive due to its dictionary lookup. To illustrate, the words "cars" and "caring" are both reduced to the stem "car", while lemmatization reduces "cars" into "car" and "caring" into "care". As a result, stemming reduces the dimensionality in a corpus more than lemmatization. Dimensionality reduction saves memory space and computation time (Asghar et al., 2014; Hickman et al., 2020; Mhatre et al., 2017). However, errors might occur during analysis because a stemmer can make the same Stem of semantically distinct words (Hickman et al., 2020). Although none of the stemmers give 100% output, they are appropriate for natural language processing applications. Stemming methods have a lot in common but it might be possible that one of them scores better in a specific application (Jivani et al., 2011).

Preprocessing methods such as stemming and lemmatization are part of the pipeline of many improved classification models in affective systems (Babanejad et al., 2020). However, there is an inconsistency in the stemmers being used by prior work. Table 1 provides an overview of the methods used in related work. The Porter stemmer and the Snowball stemmer are used in emotion classification, movie reviews, sarcasm detection, news, and e-mail. The Lovins stemmer is used by Angiani et al. (2016) for Twitter sentiment analysis. Hence, it is unclear which stemmer is most suitable. The characteristics of the stemming methods are discussed in section 3.

In addition, while preprocessing methods are well-studied in affective systems, little work has been done when applied to the training phase of word embedding models (Babanejad et al., 2020). According to Babanejad et al. (2020), most models of affect analysis make use of pretrained word embeddings. Pretrained word embedding models are trained on a large corpus. To illustrate, Google's pretrained Word2Vec model is trained on a part of the Google News Dataset, which consists of approximately 100 billion words. The actual model contains 300-dimensional vectors for three million words (Mikolov, Chen, Corrado, & Dean, 2013).

Paper	Method(s)	Application(s)
Agrawal and An 2012	Snowball stemmer	Emotion
Angiani et al. 2016	Lovins stemmer	Twitter
Babanejad et al. 2020	Snowball stemmer	Movie reviews Twitter Emotion Sarcasm
Bakliwal et al. 2012	Porter stemmer	Twitter
Bao et al. 2014	Unkown stemmer Lemmatization	Twitter
Camacho-Collados and Pilehvar 2017	Lemmatization	Topic categorization Movie reviews
Danisman and Alpkocak 2008	Porter stemmer	Emotion
Mulki et al. 2018	Snowball stemmer Lemmatization	Emotion
Symeonidis et al. 2018	Porter stemmer Lemmatization	Twitter
Uysal and Gunal 2014	Porter stemmer	News E-mail

TABLE 1: Stemming and lemmatization methods used in related work

To address this limitation, Babanejad et al. (2020) performed an extensive analysis of preprocessing methods and measured their effect when integrating earlier in word embedding models. Hence, instead of only applying preprocessing on the classification dataset, preprocessing is also applied before training the word embeddings. The results of the study showed that incorporating preprocessing into the training phase appears to be more beneficial than applying them on the classification dataset (Babanejad et al., 2020).

This thesis builds upon the work of Babanejad et al. (2020) by adding stemming and lemmatization methods that have not been employed in their work. Therefore, the goal of this research is to examine the effectiveness of stemming and lemmatization when applied independently to the training corpus before training the word embeddings. This makes it possible to compare stemming and lemmatization methods with each other and other preprocessing methods investigated by Babanejad et al. (2020). This leads to the following research question:

> *Which stemming or lemmatization method is most suitable for predicting sentiment polarity when integrated at the training phase of word vector based models?*

To address the research question, the architecture of Babanejad et al. (2020) is used. Stemming and lemmatization methods are used to preprocess the training corpus. Then, the word embedding method Word2Vec (Mikolov et al., 2013) is trained on the training corpus. The classification dataset is preprocessed in the same way as the training corpus in order to obtain a compatible vocabulary. A long short-term memory (LSTM) model with embedding layer is used to classify sentiment polarity. The model is evaluated in terms of weighted F-score. This architecture is further explained in section 4.

The experiment has been carried out four times. The results reported in this research show that all the methods outperformed the baseline for both models. For the CBOW model, the Porter stemmer, Lancaster stemmer and lemmatization achieved F-scores of 85.67%, 85.21%, and 86.00%, respectively. Babanejad et al. (2020) yielded a F-score of 83.99% for the baseline, while the Snowball stemmer had an F-score of 86.92%. However, a F-score of 85.86% is achieved when reproducing the results of the Snowball stemmer. This indicates that lemmatization is the best performing method in terms of F-score. For the Skip-gram model, the Porter stemmer, Lancaster stemmer, and lemmatization achieved F-scores of 85.90%, 83.10%, and 85.89%, respectively. Babanejad et al. (2020) produced a F-score of 83.07% for the baseline and a F-score of 86.00% for the Snowball stemmer. The Snowball stemmer had a F-score of 86.18% when reproducing the results. Hence, the Snowball stemmer is preferred method.

Babanejad et al. (2020) claim that their analysis is the first that provides insight of preprocessing methods when applied at the training phase of a word

embedding model. Only the Snowball stemmer is investigated in their work. Therefore, the contribution of this thesis is twofold. First, the impact of different stemming methods are evaluated, which makes it possible to compare them with each other. Second, the impact of lemmatization is examined. This makes it possible to compare the stemming methods with lemmatization.

This work could be beneficial from both societal and scientific points of view. Millions of people across different countries share their opinion about topics of interest, which makes sentiment analysis a commercially attractive field (Acosta, Lamaute, Luo, Finkelstein, & Andreea, 2017; Alam & Yao, 2019; Go et al., 2009). The results of this thesis give insight in the impact of stemming and lemmatization when applied before training a word embedding model. The outcome could be valuable for future research as well. Researchers can motivate their choice for stemming and lemmatization, and use the preferred method in their own research.

The remainder of this paper is organized as follows: section 2 discusses the related work. Section 3 provides information about the stemming methods, lemmatization, word embedding and classification methods. In section 4, the experimental setup is described. Section 5 presents the results of this study. Finally, section 6 provides the discussion and conclusion.

2 RELATED WORK

2.1 *Stemming and lemmatization*

Natural language texts often contain different variants of a word. It is also possible that words are misspelled, abbreviated or have alternative spellings (Willett, 2006). Stemming and lemmatization are approaches to handle inflectional forms and derivationally related forms of a word by reducing them to their root form. Stemming has its roots in the domain of information retrieval. The main idea is that the effectiveness of searching should be increased in terms of recall by conflating multiple variants of a word in order to retrieve documents, instead of only the specific variant of the search query (Jivani et al., 2011; Willett, 2006). As explained in section 1, stemming and lemmatization are practically the same except that lemmatization takes the part-of-speech and the context of the word into account (Jivani et al., 2011).

Two types of errors can occur during the stemming process, namely under-stemming and over-stemming. In under-stemming, two words that should be stemmed into the same Stem are actually stemmed into different Stems. On the contrary, over-stemming is when two words are stemmed into the same Stem where they should be stemmed into different Stems (Balakrishnan & Lloyd-Yemoh, 2014; Jivani et al., 2011).

Multiple flavors of stemming algorithms exist. The first stemmer was developed by Lovins in 1968 (Lovins, 1968). The Porter stemmer (Porter, 1980), Lancaster stemmer (Paice, 1990), and Snowball stemmer (Porter, 2001) followed. Kazmaier and van Vuuren (2020) investigated the impact of the Porter stemmer, Snowball stemmer, Lancaster stemmer, and lemmatization build upon WordNet on the vocabulary size. Out of a corpus of 5,385 words, the Lancaster stemmer appeared to be the most aggressive stemmer by yielding a vocabulary size of 3,484 words. The Porter stemmer and Snowball stemmer yielded almost the same vocabulary size with 3,951 and 3,920 words, respectively. Lemmatization only reduced the vocabulary size to 4,948.

In recent years, stemming and lemmatization have been applied to other natural language processing applications such as sentiment analysis, emotion classification and sarcasm detection (Babanejad et al., 2020). Stemming and lemmatization are also frequently used in foreign languages such as Indonesian (Adriani, Asian, Nazief, Tahaghoghi, & Williams, 2007), Arabic (Aljlayl & Frieder, 2002; Kadri & Nie, 2006), and French (Savoy, 1993), to name a few.

2.2 *Preprocessing used in machine learning methods*

Pang, Lee, and Vaithyanathan (2002) introduced the first approach to classify sentiment of movie reviews with machine learning classification algorithms. A bag-of-words approach is used to create feature vectors. Naïve Bayes (NB), Maximum Entropy (MaxE), and Support Vector Machines (SVM) classifiers were used for classification. The authors concluded that all the classifiers outperformed human-produced baselines based on indicator words with about 20%. A unigram approach was the basis of the best performances. The SVM performed best with an accuracy score of 82.9%, followed by the NB and MaxE with scores of 81.5% and 81%, respectively.

Go et al. (2009) showed that the same results can be yielded when classifying polarity of Twitter messages using the same machine learning classifiers as Pang et al. (2002). Data is collected manually through an Application Programming Interface (API). Then, they reduced the feature space by replacing any mention (i.e. "@") with the tag "USERNAME" and any link with the tag "URL". Also, letters that are repeated more than two times in a row are replaced by two occurrences. Usage of unigram and unigram + bigram resulted in the best results. The best result was yielded by MaxE, followed by NB and SVM with 83%, 82.7%, and 82.2%, respectively.

Bakliwal et al. (2012) tested the effect of preprocessing methods individually on the accuracy of a SVM classifier. Multiple preprocessing methods including stopwords removal, the Porter stemmer, and spell correction were one by one employed to measure their impact on accuracy scores on two different datasets, including the sentiment140 dataset created by Go et al. (2009). Each

preprocessing filter is compared to a baseline which consisted of basic cleaning operations such as the removal of special characters, URLs, mentions, emoticons, and hashtags. The authors used a NB method to determine the polarity of a token. Then, the positive and negative probabilities of all tokens in a tweet are added and their difference is calculated. If the score is > 0, the tweet is positive. Otherwise it is negative. The results show that each preprocessing method improved the accuracy score on both datasets. For the sentiment140 dataset, baseline + stemming achieved the best result with an accuracy score of 81.9%, whereas baseline + emoticons + punctuations yielded the highest accuracy score of 80.3% on the Mejaj dataset.

Bao, Quan, Wang, and Ren (2014) investigated the individual impact of multiple preprocessing methods on the sentiment140 Twitter dataset created by Go et al. (2009). The effects of repeated letters, negation, URLs, stemming and lemmatization were examined using WEKA's Liblinear logistic regression. The authors used unigrams and a combination of unigrams and bigrams in order to structure the feature space. Term frequency, information gain, and chi-square statistics were applied to select bigram features. Preprocessing methods were compared to a baseline consisting of some basic cleaning steps such as the removal of usernames, hashtags, and punctuations. First, stopwords and negation handling were added towards the baseline. Accuracy improved by 0.28% as a result. The impact of negation is much bigger. The accuracy score improved with 2.78%. Hereafter, repeated letter handling was added resulting in an increase of 0.28%. Finally, stemming and lemmatization were added to the pipeline. The accuracy score decreased for both methods with 3.07% and 3.32%, respectively.

Preprocessing methods for Twitter sentiment analysis are also compared individually by Angiani et al. (2016) on the SemEval 2015 and SemEval 2016 datasets. A selection of preprocessing methods, namely stemming, stopwords, negation, emoticon, and a dictionary of slang words were investigated. First, basic cleaning operations including the removal of URLs, hashtags, mentions, punctuations, and extra blank spaces are applied. After the cleaning process, the authors applied preprocessing methods individually and measured their accuracy using a Naïve-Bayes Multinomial classifier. Only the dictionary of slang words did not improve accuracy. It turned out that basic cleaning operations in combination with stemming resulted in the highest accuracy score. An accuracy score of 68,68% was achieved on the test set of 2016, while an accuracy score of 76.4% was obtained on the test set of 2015. This means that the accuracy score increased by 3.28% and 2.32% compared to the baseline, respectively. The authors concluded that accuracy increased because the stemming methods equate several word variants to their root form. As a result, words can be selected as useful features by the classifier.

Uysal and Gunal (2014) studied the impact of tokenization, the Porter stemmer, lowercasing and stopword removal on the accuracy of a Support Vector Machine in two different domains, namely news and e-mail. Features were selected using the chi-square method. All possible combinations of the four preprocessing methods were considered to reveal possible interactions. The authors concluded that stemming is required for both domains. For the e-mail domain, which is a binary classification problem, the best combination is tokenization, lowercasing, and stemming. This combination achieved a Micro-F1 score of 98.88%. For the news domain, which is a multi-class classification problem, a combination of lowercasing and stemming yielded the best performance with a Micro-F1 score of 87.19%.

Danisman and Alpkocak (2008) investigated the effect of stemming in classifying emotion on the SemEval 2007 dataset. The authors used Naïve Bayes (NB), Support Vector machines (SVM), and Vector Space Model (VSM) for classification. The authors found that stemming removes the emotional meaning from words. For instance, the word 'marry' is classified as joy whereas 'married' is classified as sad. However, these samples are rare in the test sets. As a result, stemming increased classification performance for all classifiers. Accuracy scores for NB, SVM, and VSM increased with 4.5%, 0.8%, and 4%, respectively. F1 scores are measured as well. The F1 scores increased with 3.8%, 2.1%, and 2.6%, respectively.

Agrawal and An (2012) also evaluated the effect of stemming in predicting emotion on two different corpora, namely Wikipedia and Gutenberg. Stemming did improve the accuracy scores with 5.28% and 0.58%, respectively. The authors suggested that although stemming reduces emotional meaning of words, the roots of words are related close enough to a particular emotion.

Mulki, Ali, Haddad, and Babaoğlu (2018) investigated the impact of preprocessing in multi-label emotion classification. The authors tackled task 1 in the SemEval-2018 contest, which requires to classify one or more of 11 emotion labels embedded in Tweets for the English, Arabic, and Spanish language. The multi-label classification (MLC) problem is transformed into binary classification problems using Binary Relevance (BR). Hereafter, a term frequency-inverse document frequency weighting scheme is applied to create feature vectors. A Multi-label SVM classifier (one-Vs-All) with linear kernel is used for classification. Finally, an amount of binary SVMs equally to the number of labels were trained to recognize emotions in a Tweet. Stemming improved the accuracy score by 5.1% compared to removing stopwords in Arabic. For English and Spanish, stemming improved the accuracy by 0.3% and 0.8%, respectively. For Arabic, the best combination is emoticon tagging, stemming, and stopwords removal. This combination achieved an accuracy score of 44.9%. For both English and Spanish, emoticon tagging, lemmatization, and stopwords removal achieved the best performance. Accuracy scores of 48% and 43.1%

were obtained for English and Spanish, respectively. The authors concluded that lemmatization is a better choice compared to stemming for the English language, because lemmatization is better in handling informal Tweets due to its implicitly parts-of-speech tagging.

To conclude, stemming and lemmatization improve classification performance in many occasions across multiple applications such as Twitter (Angiani et al., 2016; Bakliwal et al., 2012), emotion classification, (Agrawal & An, 2012; Danisman & Alpkocak, 2008; Mulki et al., 2018), news, and e-mail (Uysal & Gunal, 2014). However, prior work used different stemming algorithms, which is indicated in table 1. Also, prior work lacks a comparison of different stemming algorithms. Therefore, it would be worthwhile to investigate if there is a stemming method which yields the best performance.

2.3 *Preprocessing used in Neural Networks*

Camacho-Collados and Pilehvar (2017) investigated the role of lemmatization, lowercasing and multiword grouping on the performance of a convolutional neural network (CNN). Their work was motivated because text preprocessing in deep learning has not received much attention. A total of nine datasets over two tasks (topic categorization and sentiment polarity detection) were evaluated. The preproccesing methods were compared to a vanilla model which consists of only the tokenized corpus. The authors concluded that a simple tokenized corpus achieves a similar or even higher accuracy score than lemmatization and multiword grouping on non domain-specific datasets. Zooming in on sentiment analysis, the highest accuracy scores were obtained when an embedding layer (pretrained Word2Vec (Mikolov et al., 2013)) and LSTM layer were included in the model. To illustrate, an accuracy score of 88.9% was achieved on the IMDB dataset when using only a tokenized corpus.

Symeonidis et al. (2018) performed an extensive comparison of 16 commonly preprocessing methods including the Porter stemmer and lemmatization for two Twitter sentiment analysis datasets. Four algorithms, namely Logistic Regression, Linear SVC, Bernoulli Naïve Bayes, and a Convolutional Neural Network have been employed for classification. In general, the authors concluded that lemmatization, replacing contractions, and removing numbers improved model accuracy. For the SS-Twitter dataset, the best performing methods are replacing contractions, remove numbers, replace repetitions of punctuations, lemmatization, stemming, and handling negations. For the SemEval dataset, the best performing methods are replacing contractions, remove numbers, replace repetitions of punctuations, and the removal of stopwords.

When zooming in on stemming and lemmatization, lemmatization outperformed the baseline results in both datasets. For the SS-Twitter dataset, accuracy improved with 0.3% for Logistic regression, 0.8% for Bernoulli Naïve

Bayes, 0.2% for Linear SVC. However, accuracy decreased with 1.5% for the CNN. For the SemEval dataset, accuracy scores increased with 0.2% and 0.3% for Logistic Regression and Bernoulli Naïve Bayes, respectively. Accuracy scores decreased with 0.7% and 2.2% for Linear SVC and CNN, respectively. Stemming only outperformed the baseline on the SS-Twitter dataset. Accuracy scores increased for the Logistic Regression, Bernoulli Naïve Bayes, and Linear SVC with 1.1%, 2.7%, and 0.1%, respectively. The accuracy decreased with 2% for the CNN. Stemming performed worse on the SemEval dataset. All the accuracy scores decreased, namely with 0.1% for Logistic Regression, 0.1% for Bernoulli Naïve Bayes, 0.9% for Linear SVC, and 3.9% for CNN, respectively.

Babanejad et al. (2020) claims to be the first that investigated preprocessing in word vector based models. Their work was motivated by the fact that preprocessing methods are part of nearly every improved text classification model. Spellcheck, negation, parts-of-speech, stopwords, and stemming were investigated. The word embedding methods CBOW and Skip-gram from Word2Vec (Mikolov et al., 2013) and BERT-large (Devlin, Chang, Lee, & Toutanova, 2018) were trained from scratch on two different training corpora, namely News and Wikipedia. Then, the word vectors are used in a LSTM model on nine classification datasets over three different affective tasks (sentiment analysis, emotion classification, and sarcasm detection). Performance was measured in terms of weighted f-score.

First, the effect of each preprocessing method is investigated when applied at the training phase of the word vector model. This is only carried out for CBOW and Skip-gram on the News corpus. The authors concluded that only a single preprocessing method improves the performance of the LSTM model. Negation was the best performing method on all datasets for both CBOW and Skip-gram followed by parts-of-speech. The other methods yielded mixed results, but still performed better than the baseline on all sentiment analysis datasets. Also, an ablation study (i.e. all methods minus one) has been carried out in order to find useful combinations. For CBOW, the best results were obtained with "all - stop" combination. The best combination for Skip-gram is either "all - stop" or "all - stem", depending on the dataset. For the Wikipedia corpus, which is about 64 times bigger than the News corpus, only an ablation study has been carried out. The best results were achieved with "all - stem" for CBOW, while "all - stop" yielded the best performance for both Skip-gram and Bert large.

Second, the difference between applying preprocessing methods only on the training corpus and applying preprocessing methods only on the classification dataset has been examined for the Wikipedia corpus. The authors concluded that incorporating preprocessing into the training corpus before generating word vectors outperforms preprocessing on the classification datasets.

Finally, the authors compared their best result for each of the three word embedding methods (i.e. BERT-large "all - stop") with several pretrained word

embedding models such as GloVe, SSWE, and FastText. BERT achieved the best result on eight out of nine datasets. Word2Vec's CBOW is the second best on four datasets.

To conclude, the work of Babanejad et al. (2020) showed that incorporating preprocessing methods in training corpora is more beneficial than applying them on classification datasets. The best performing method is "all - stop". The authors suggest that exploring the space of preprocessing methods might yield in interesting results. Hence, this thesis builds upon their work by investigating the effectiveness of stemmers and lemmatization available in Python's NLTK library. These are the Porter stemmer, Lancaster stemmer and Wordnet Lemmatizer (Bird, Klein, & Loper, 2009). The Snowball stemmer is also available in the NTLK library, but is already employed in the work of Babanejad et al. (2020).

3 METHODS

3.1 *Porter stemmer*

The Porter stemmer is considered as a simplistic truncating stemmer proposed by Porter (1980). The main idea is to remove common morphological and inflectional endings. The algorithm applies a set of five algorithmic steps with corresponding rules in order to decide to remove the suffix of a word (Porter, 1980). The rules are utilized in the following order (Patil & Patil, 2013; Porter, 1980):

1. Plurals and participles handling (i.e. ponies –> poni).
2. Patterns are matched on the same common suffixes (i.e. conditional –> condition).
3. Special word endings handling (i.e. formalize –> formal).
4. Check the stripped word again for suffixes in case the word is compounded (i.e. revival –> reviv)
5. Check whether the stripped word ends with a vowel. If yes, fix appropriately (i.e. probate –> probat).

Compared to other stemmers, the Porter stemmer returns the best output and has less error rate (Jivani et al., 2011). The Porter stemmer only supports the English language (Bird et al., 2009).

3.2 *Lancaster stemmer*

The Lancaster (or Paice/Husk) stemmer is an iterative algorithm developed in 1990 and is frequently used at Lancaster University (Paice, 1990). This stemming

algorithm has a set of 120 rules. The result of each rule is either a deletion or replacement of an ending, and is executed on the final letter of a suffix in every iteration. The Lancaster stemmer terminates in three occasions. First, when none of the rules can be applied. Second, if a word starts with a vowel and only has two letters left. Finally, if a word starts with a consonant and there are only three characters left (Jivani et al., 2011; Paice, 1990). The Lancaster stemmer is considered as an aggressive stemmer since it continuously applies its rules until it terminates. Hence, it could be the case that words are not meaningful due to over-stemming (i.e. destabilize –> dest) (Benghuzzi & Elsheh, 2020; Jivani et al., 2011).

3.3 *Lemmatization*

In contrast to stemming, lemmatization requires a supportive dictionary for indexing in order to return a valid word in language (Asghar et al., 2014). Lemmatization gives more accurate results for inflection removal compared to stemming, because it takes the part of speech and the context of the word into account (i.e. runs, running, ran –> run). However, it is a more time-consuming process (Asghar et al., 2014; Mhatre et al., 2017). WordNet is a dictionary with 155,287 words and 117,659 synonym sets. The WordNet Lemmatizer only removes affixes if the word is in the dictionary. If the word cannot be found in the dictionary, it returns the input unchanged. (Bird et al., 2009).

3.4 *Word2Vec*

Word2Vec is a word embedding method developed by Mikolov et al. (2013). It can be trained on a corpus of words and produces a vector for each unique word in the corpus. Words that occur frequently in similar linguistic contexts are positioned close to each other in the embedding space. The output of the network is a vector with probability scores that a randomly selected nearby word is that vocabulary word (Babanejad et al., 2020; Mikolov et al., 2013).

The Word2Vec method has two architectures: Continuous Bag-Of-Words (CBOW) and continuous Skip-gram (SG). CBOW and Skip-gram are the opposites of each other. CBOW predicts the target word from its neighbouring words, whereas Skip-gram predicts potential neighbouring words based on the single word being analyzed (Mikolov et al., 2013).

3.5 *Long short-term memory*

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture developed by Hochreiter and Schmidhuber (1997). LSTM enables

the hidden state of an RNN to span over long sequences and as a result prevent the vanishing gradient problem (Hochreiter & Schmidhuber, 1997; Xiao & Cho, 2016).

The main intuition of LSTM is its cell state and gates. The cell state refers to the memory of the network and transfers information during the entire sequence. Throughout the processing, the gates add or remove information from the cell state. First, the forget gate decides what information is relevant from the previous steps. Second, the input gate decides what information should be added from the current step. Finally, the output gate decides what the next hidden state is (Hochreiter & Schmidhuber, 1997; Xiao & Cho, 2016).

4 EXPERIMENTAL SETUP

4.1 *Data*

Two datasets are used in this thesis, namely the News corpus and the IMDB dataset. The News corpus consists of 142,570 articles from 15 American publications. The news articles were collected from 2013 to 2018. The dataset is publicly available in CSV format on Kaggle[1].

The classification dataset used for this thesis is the IMDB movie reviews dataset created by Maas et al. (2011). The dataset consists of 25k positive reviews and 25k negative reviews and can be obtained in CSV format on Kaggle[2]. The dataset consists of only two columns, namely review and sentiment. No more than 30 reviews per movie are included in the dataset. Sentiment polarity is based on review scores on a scale from 1 out of 10. Negative reviews have a score of less than 4 while positive reviews have a score of at least 7 (Maas et al., 2011).

4.2 *Training corpus setup*

Word vector models are trained on the News corpus. First, some basic preprocessing operations are applied on the dataset. These include the removal of common punctuations, numbers, extra space, HTML tags, and lowercasing. Then, words are tokenized using NLTK's word_tokenize (Bird et al., 2009). This is considered as the baseline in the work of Babanejad et al. (2020). Second, the tokens are stemmed using NLTK's PorterStemmer and LancasterStemmer. Also, NLTK's WordNetLemmatizer is used to create Lemma's (Bird et al., 2009). Finally, CBOW and Skip-gram models are created for every stemming method and lemmatization using the gensim.models package (Srinivasa-Desikan, 2018).

[1] https://www.kaggle.com/snapcrack/all-the-news/code
[2] https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

This means that we have a total of six models. Each model has the same parameter settings, namely a minimum word count of 5, vector size of 300 and a window of 5. These parameters are used in the work of Babanejad et al. (2020).

4.3 *Classification setup*

The IMDB dataset is preprocessed in the same way as in section 4.2 in order to have a compatible vocabulary. Sentiment labels are recoded such that negative tweets have the label 0 and positive tweets have the label 1. Train and test sets are created using the train_test_ split function from the scikit-learn machine learning library (Pedregosa et al., 2011). The ratio between training and testing is 80% training (with 10% validation) and 20% test. For classification, a LSTM model from the Keras library is used (Chollet et al., 2015). The network starts with an embedding layer, followed by a LSTM layer of 128 hidden units. Finally, the output layer is a Dense layer of a single-unit with a sigmoid activation function. The Adam optimizer (Kingma & Ba, 2014) is used along with binary cross-entropy as loss function. Finally, the model has a batch size of 128 and runs for 6 epochs. Model performance is reported in terms of weighted F-score (Pedregosa et al., 2011). We used the source code of the authors[3]. Close inspection revealed that the source code deviated from the method reported in the paper in the following ways: basic preprocessing, batch size, and epochs. The above classification setup is used, because this setup yielded the closest results in terms of reproducibility. The code used in this research can be obtained via Github[4].

5 RESULTS

5.1 *CBOW model*

This section provides the experimental results of the CBOW model. First, an overview of the amount of unique words in the training corpus and word embedding models are presented in table 2. A total of 111,003,223 words are in the news corpus of which 118,561 are unique words. The reduction of the number of unique words is in line with the work of Kazmaier and van Vuuren (2020). The Lancaster stemmer is the most aggressive stemming algorithm by reducing the amount of unique words to 88,340. The Porter stemmer and Snowball stemmer have almost an equal amount of unique words with 100,394 and 100,383 words, respectively. Lemmatization slightly reduces the amount of unique words. The amount of unique words is reduced to 115,609 words.

[3] https://github.com/NastaranBa/preprocessing-for-word-representation
[4] https://github.com/ysenders/DSS_thesis

A word is incorporated in the Word2Vec model if it occurred at least five times. When no stemming or lemmatization is applied, the Word2Vec models have a vocabulary of 113,916 words. The Lancaster stemmer has a vocabulary size of 69,697, which is by far the smallest. Again, the Porter stemmer and Snowball stemmer have comparable vocabulary sizes of 82,639 and 82,273 words, respectively. Finally, lemmatization has a vocabulary size of 103,007 words. These vocabulary sizes are the same for both CBOW and Skip-gram.

Method	News corpus	Word embedding model
No stemming or lemmatization	118,561	113,916
Porter stemmer	100,394	82,639
Lancaster stemmer	88,340	69,697
Lemmatization	115,609	103,007
Snowball stemmer	100,383	82,273

TABLE 2: Number of unique words in training corpus and word embedding model by method

The baseline of Babanejad et al. (2020) achieved an F-score of 83.99% whereas the Snowball stemmer yielded an F-score of 86.92%.

Method	Accuracy	Precision	Recall	F-score
Porter stemmer	85.41	**84.87**	86.64	85.67
Lancaster stemmer	84.70	83.19	87.39	85.21
Lemmatization	**85.49**	83.73	**88.45**	**86.00**

TABLE 3: Mean accuracy, precision, recall, and weighted F-score percentages by method for the CBOW model (N=4).

Table 3 displays the classification performance in terms of accuracy, precision, recall, and weighted F-score for each method. The experiment has been carried out four times. Hence, mean scores are presented. The Porter stemmer achieved an F-score of 85.47%. The Lancaster stemmer is the worst performing method with an F-score of 85.21%. On the contrary, lemmatization is the best performing method of this study with an F-score of 86.00%. If we compare the F-scores with the baseline and Snowball stemmer in the work of Babanejad et al. (2020), it can be concluded that the each method outperformed the baseline. However, the Snowball stemmer still has the best performance across all methods.

Table 4 shows the performance of the Snowball stemmer when trying to reproduce the results of Babanejad et al. (2020). Mean percentages are also presented based on four runs. The mean F-score of the Snowball stemmer is

Method	Accuracy	Precision	Recall	F-score
Snowball stemmer	85.45	84.17	87.67	85.86

TABLE 4: Mean accuracy, precision, recall, and weighted F-score percentages for the Snowball stemmer for the CBOW model (N=4).

85.86%. This is about 1% lower than the score in the work of Babanejad et al. (2020) while running the same script four times. However, if we compare the results of the Snowball stemmer in table 4 with the results of the other methods, the Snowball stemmer is the second best performing method. Although the scores are close to each other, lemmatization is the best performing method in terms of F-score.

When comparing the other evaluation metrics from table 3, the Porter stemmer achieved an accuracy score of 85.41%, precision score of 84.77%, and a recall score of 86.64%. The Lancaster stemmer yields scores of 84.70%, 83.19%, and 87.39%, respectively. Lemmatization had scores of 85.49%, 83.73%, and 88.45%, respectively. Finally, the snowball stemmer in table 4 achieved an accuracy score of 85.45%, a precision score of 84.17%, and a recall score of 87.67%. Hence, lemmatization has the best performance in terms of accuracy, recall and F-score. The Porter stemmer achieved the highest precision score. The results for each run of the experiment are in Appendix A (page 25).

Confusion matrices from one run of the experiments are shown in table 5. In general, stemming and lemmatization have more false negatives (FN) than false positives (FP). Only the Porter stemmer has more FP than FN in this run. However, the mean recall score is higher than the mean precision score. This is in line with the literature, which indicated that the effectiveness of searching should be increased in terms of recall (Jivani et al., 2011; Willett, 2006).

Finally, the F-score of the best performing method lemmatization can be compared to the other preprocessing methods in the work of Babanejad et al. (2020). Out of 7 different preprocessing methods, lemmatization is placed fifth. However, we were unable to reproduce the exact same results of the Snowball stemmer. Therefore, lemmatization might climb because it actually produced better results than the Snowball stemmer in this thesis.

5.2 *Skip-gram model*

This section provides the experimental results of the Skip-gram model. The baseline of Babanejad et al. (2020) achieved an F-score of 83.07%, whereas the Snowball stemmer yielded an F-score of 86.00%.

Table 7 shows the mean classification performance in terms of accuracy, precision, recall, and weighted F-score for each method. The experiment has

(A) CM of Lancaster stemmer

actual \ Predicted	0	1
0	TN 4046	FP 915
1	FN 554	TP 4485

(B) CM of Lemmatization

actual \ Predicted	0	1
0	TN 4160	FP 801
1	FN 653	TP 4386

(C) CM of Porter stemmer

actual \ Predicted	0	1
0	TN 4354	FP 607
1	FN 867	TP 4172

(D) CM of Snowball stemmer

actual \ Predicted	0	1
0	TN 3967	FP 994
1	FN 504	TP 4535

TABLE 5: Confusion matrices of CBOW models

been carried out four times. The Porter stemmer achieved an F-score of 85.90%. The Lancaster stemmer is the worst performing method with an F-score of 83.10%. Finally, lemmatization had a F-score of 85.89%. If we compare the F-scores with the baseline and Snowball stemmer in the work of Babanejad et al. (2020), it can be concluded that each method outperformed the baseline again. However, the Snowball stemmer is still the best performing method in terms of F-score.

Table 8 provides the performance of the reproduced Snowball stemmer from the work of Babanejad et al. (2020). Percentages are also presented based on the mean of four runs. The Snowball stemmer achieved an F-score of 86.18%. In contrast to the CBOW model, the F-score of the Snowball stemmer for the Skip-gram model is much closer. Regardless of which F-score is taken, both Snowball stemmer scores performed better than the methods in table 7.

When comparing the other evaluation metrics from table 7, the Porter stemmer achieved an accuracy score of 85.86%, precision score of 86.33%, and a recall score of 85.57%. The Lancaster stemmer yields scores of 82.09%, 80.57%,

Method	Accuracy	Precision	Recall	F-score
Porter stemmer	85.86	86.33	85.57	85.90
Lancaster stemmer	82.09	80.57	87.42	83.10
Lemmatization	**86.00**	**87.26**	84.72	85.89

TABLE 7: Mean accuracy, precision, recall, and weighted F-score percentages by method for the Skip-gram model (N=4).

Method	Accuracy	Precision	Recall	F-score
Snowball stemmer	85.77	84.50	**88.05**	**86.18**

TABLE 8: Mean accuracy, precision, recall, and weighted F-score percentages for the Snowball stemmer for the Skip-gram model (N=4).

and 87.42%, respectively. This is by far the worst performing method. Lemmatization had scores of 86.00%, 87.26%, and 84.72%, respectively. Finally, the snowball stemmer from table 8 achieved an accuracy score of 85.77%, a precision score of 84.50%, and a recall score of 88.05%. To conclude, lemmatization has the best performance in terms of accuracy and recall while the Snowball stemmer produced the highest recall and F-score.

Confusion matrices of one run of the experiment for each method is presented in table 9. The stemming methods have more FP than FN, which indicates that the recall score is higher. Remarkably, the Lancaster stemmer has very few FN. The precision score of lemmatization is higher than the recall score, which is in line with the mean precision score and mean recall score.

Finally, we can compare the F-score of lemmatization with the other preprocessing methods in the work of Babanejad et al. (2020). Out of 7 preprocessing methods, lemmatization gets a sixth place. However, spellcheck and parts-of-speech scored slightly higher with 85.90%, and 85.91%, respectively.

6 DISCUSSION AND CONCLUSION

6.1 *Findings*

The goal of this study was to investigate the effectiveness of different stemming and lemmatization algorithms when applied at the training phase of a word vector model. To address the research question "Which stemming or lemmatization method is most suitable for predicting sentiment polarity when integrated at the training phase of word vector based models?", the architecture of Babanejad et al. (2020) is used. The Porter stemmer, Lancaster stemmer, and

(A) CM of Lancaster stemmer

actual \ Predicted	0	1
0	TN 3316	FP 1645
1	FN 281	TP 4758

(B) CM of Lemmatization

actual \ Predicted	0	1
0	TN 4381	FP 580
1	FN 846	TP 4193

(C) CM of Porter stemmer

actual \ Predicted	0	1
0	TN 4180	FP 781
1	FN 592	TP 4447

(D) CM of Snowball stemmer

actual \ Predicted	0	1
0	TN 4120	FP 841
1	FN 584	TP 4455

TABLE 9: Confusion matrices of Skip-gram models

lemmatization were examined and compared to the Snowball stemmer which was already employed in the work of Babanejad et al. (2020). Word2Vec's CBOW and Skip-gram architectures were used to create word embeddings which were trained on the News corpus. A LSTM model with embedding layer is used to predict sentiment polarity on the IMDB movie reviews dataset. The experiments have been carried out four times. Mean percentages were reported in this study.

In the case of the CBOW model, lemmatization yielded a F-score of 86.00% and outperformed the Porter stemmer and Lancaster stemmer. These stemmers had F-scores of 85.67% and 85.21%, respectively. Although lemmatization performed worse than the Snowball stemmer in the work of Babanejad et al. (2020), it yielded better results when the results were reproduced using the same script. The Snowball stemmer had an F-score of 86.92% in the work of Babanejad et al. (2020). However, it achieved a mean F-score of 85.86% when reproducing the script. Also, the results were systematically lower each run. The authors did not specify how many times the experiment has been carried

out. A possible reason could be that the the authors only ran the experiment once and obtained a very high performance.

The results for the Skip-gram model showed that the Snowball stemmer is the best performing method in terms of F-score, even when reproducing the results. The Snowball stemmer in the work of Babanejad et al. (2020) produced a F-score of 86.00%. When reproducing the results, the Snowball stemmer achieved a F-score of 86.18%. The Porter stemmer, Lancaster stemmer and lemmatization yielded F-scores of 85.90%, 83.10% and 85.89%.

The Lancaster stemmer was the worst performing method for both models, especially in the case of the Skip-gram model. The Lancaster stemmer had the smallest vocabulary size, which might be a result of either over-stemming or by making the same Stem of many different words.

To conclude, the results of the stemming and lemmatization methods are quite close to each other in terms of F-score except for the Lancaster stemmer. However, lemmatization and the Snowball stemmer yielded the best results. This is also the case for other evaluation metrics including accuracy, precision, and recall.

6.2 *Implications*

While preprocessing methods such as stemming and negation are part of the pipeline of many improved text classification methods, little work has been carried out when applied at the training phase of word vector models in affective systems. This was the main reason that motivated Babanejad et al. (2020) to perform a comprehensive analysis of preprocessing methods in word vector based models. This thesis builds upon their work by investigating different preprocessing methods, namely the Porter stemmer, Lancaster stemmer, and lemmatization when applied to Word2Vec's CBOW and Skip-gram architectures. Therefore, the contribution of this thesis to existing literature is twofold. First, the impact of different stemming methods are evaluated, which makes it possible to compare them with each other. Second, the impact of lemmatization is examined, which is a new method in the spectrum of preprocessing methods in the work of Babanejad et al. (2020). This makes it possible to compare the stemming methods with lemmatization.

6.3 *Limitations and future work*

The work of Babanejad et al. (2020) was extensive. Because of limited time and computational resources, it was not feasible to replicate the entire study for the stemming and lemmatization methods. Therefore, the following things could be done in future work.

First, this thesis showed that although close, lemmatization outperformed the Snowball stemmer in the CBOW model. It would be interesting to perform an ablation study (i.e. all - 1) in order to see how lemmatization works together with other preprocessing methods. If yes, than the best performing combination ("all - stop") would have a better performance.

Second, we only used one classification dataset. Babanejad et al. (2020) employed a total of nine datasets with different characteristics such as imbalanced classes and multi-label classification problems in their work. Therefore, it would be interesting to perform the same analysis in order to verify if the results generalize. It might be possible that one stemming algorithm is better in a specific application (Jivani et al., 2011).

Third, Babanejad et al. (2020) also used a Wikipedia corpus to train Word2Vec's CBOW and Skip-gram model. The Wikipedia corpus is a very large corpus with 8.1 billion words, which is much bigger than the News corpus that consists of 123 million words. Although the authors only performed an ablation study, it would be interesting to see if the results generalize on a much bigger corpus.

Finally, we wanted to include more state-of-the-art models such as BERT (Devlin et al., 2018). However, Babanejad et al. (2020) used a BERT-large model and explained that training a BERT model on such a large corpus as Wikipedia could take 4-5 days for each run. Reproducing that might cost more time given that they had more computational resources. Therefore, it was not feasible to train BERT-large for each stemming and lemmatization method. However, the results of Babanejad et al. (2020) showed that stemming is part of the best performing combination ("all - stop"). With this combination, the BERT-large model achieved the best performance in terms of F-score on 8 out of 9 datasets. Therefore, it would be worthwhile to investigate the stemming and lemmatization methods when training BERT from scratch.

REFERENCES

Acosta, J., Lamaute, N., Luo, M., Finkelstein, E., & Andreea, C. (2017). Sentiment analysis of twitter messages using word2vec. *Proceedings of student-faculty research day, CSIS, Pace University*, *7*, 1–7.

Adriani, M., Asian, J., Nazief, B., Tahaghoghi, S. M., & Williams, H. E. (2007). Stemming indonesian: A confix-stripping approach. *ACM Transactions on Asian Language Information Processing (TALIP)*, *6*(4), 1–33.

Agrawal, A., & An, A. (2012). Unsupervised emotion detection from text using semantic and syntactic relations. In *2012 ieee/wic/acm international conferences on web intelligence and intelligent agent technology* (Vol. 1, pp. 346–353).

Alam, S., & Yao, N. (2019). The impact of preprocessing steps on the accuracy of machine learning algorithms in sentiment analysis. *Computational and*

Mathematical Organization Theory, *25*(3), 319–335.

Aljlayl, M., & Frieder, O. (2002). On arabic search: improving the retrieval effectiveness via a light stemming approach. In *Proceedings of the eleventh international conference on information and knowledge management* (pp. 340–347).

Angiani, G., Ferrari, L., Fontanini, T., Fornacciari, P., Iotti, E., Magliani, F., & Manicardi, S. (2016). A comparison between preprocessing techniques for sentiment analysis in twitter. In *Kdweb*.

Asghar, M. Z., Khan, A., Ahmad, S., & Kundi, F. M. (2014). A review of feature extraction in sentiment analysis. *Journal of Basic and Applied Scientific Research*, *4*(3), 181–186.

Babanejad, N., Agrawal, A., An, A., & Papagelis, M. (2020). A comprehensive analysis of preprocessing for word representation learning in affective tasks. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 5799–5810).

Bakliwal, A., Arora, P., Madhappan, S., Kapre, N., Singh, M., & Varma, V. (2012). Mining sentiments from tweets. In *Proceedings of the 3rd workshop in computational approaches to subjectivity and sentiment analysis* (pp. 11–18).

Balakrishnan, V., & Lloyd-Yemoh, E. (2014). Stemming and lemmatization: A comparison of retrieval performances.

Bao, Y., Quan, C., Wang, L., & Ren, F. (2014). The role of pre-processing in twitter sentiment analysis. In *International conference on intelligent computing* (pp. 615–624).

Benghuzzi, H., & Elsheh, M. M. (2020). An investigation of keywords extraction from textual documents using word2vec and decision tree. *International Journal of Computer Science and Information Security (IJCSIS)*, *18*(5).

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Camacho-Collados, J., & Pilehvar, M. T. (2017). On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis. *arXiv preprint arXiv:1707.01780*.

Chollet, F., et al. (2015). *Keras*. https://github.com/fchollet/keras. GitHub.

Danisman, T., & Alpkocak, A. (2008). Feeler: Emotion classification of text using vector space model. In *Aisb 2008 convention communication, interaction and social intelligence* (Vol. 1, p. 53).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, *1*(12), 2009.

Hickman, L., Thapa, S., Tay, L., Cao, M., & Srinivasan, P. (2020). Text preprocessing for text mining in organizational research: Review and recommendations. *Organizational Research Methods*, 1094428120971683.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Jianqiang, Z., & Xiaolin, G. (2017). Comparison research on text pre-processing methods on twitter sentiment analysis. *IEEE Access*, *5*, 2870–2879.

Jivani, A. G., et al. (2011). A comparative study of stemming algorithms. *Int. J. Comp. Tech. Appl*, *2*(6), 1930–1938.

Kadri, Y., & Nie, J.-Y. (2006). Effective stemming for arabic information retrieval. In *Proceedings of the challenge of arabic for nlp/mt conference, londres, royaume-uni* (pp. 68–74).

Kazmaier, J., & van Vuuren, J. (2020). Sentiment analysis of unstructured customer feedback for a retail bank. *ORiON*, *36*(1), 35–71.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kotsiantis, S. B., Kanellopoulos, D., & Pintelas, P. E. (2006). Data preprocessing for supervised leaning. *International Journal of Computer Science*, *1*(2), 111–117.

Krouska, A., Troussas, C., & Virvou, M. (2016). The effect of preprocessing techniques on twitter sentiment analysis. In *2016 7th international conference on information, intelligence, systems & applications (iisa)* (pp. 1–5).

Lovins, J. B. (1968). Development of a stemming algorithm. *Mech. Transl. Comput. Linguistics*, *11*(1-2), 22–31.

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies* (pp. 142–150).

Mhatre, M., Phondekar, D., Kadam, P., Chawathe, A., & Ghag, K. (2017). Dimensionality reduction for sentiment analysis using pre-processing techniques. In *2017 international conference on computing methodologies and communication (iccmc)* (pp. 16–21).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mulki, H., Ali, C. B., Haddad, H., & Babaoğlu, I. (2018). Tw-star at semeval-2018 task 1: Preprocessing impact on multi-label emotion classification. In *Proceedings of the 12th international workshop on semantic evaluation* (pp. 167–171).

Paice, C. D. (1990). Another stemmer. In *Acm sigir forum* (Vol. 24, pp. 56–61).

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques. *arXiv preprint cs/0205070*.

Patil, C. G., & Patil, S. S. (2013). Use of porter stemming algorithm and svm

for emotion extraction from news headlines. *International Journal of Electronics, Communication and Soft Computing Science & Engineering (IJECSCSE)*, *2*(7), 9.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... others (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, *12*, 2825–2830.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*.

Porter, M. F. (2001). *Snowball: A language for stemming algorithms*.

Savoy, J. (1993). Stemming of french words based on grammatical categories. *Journal of the American Society for Information Science*, *44*(1), 1–9.

Srinivasa-Desikan, B. (2018). *Natural language processing and computational linguistics: A practical guide to text analysis with python, gensim, spacy, and keras*. Packt Publishing Ltd.

Symeonidis, S., Effrosynidis, D., & Arampatzis, A. (2018). A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis. *Expert Systems with Applications*, *110*, 298–310.

Uysal, A. K., & Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing & Management*, *50*(1), 104–112.

Willett, P. (2006). The porter stemming algorithm: then and now. *Program*.

Xiao, Y., & Cho, K. (2016). Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*.

APPENDIX A: SCORES OF EACH EXPERIMENT

Method	Accuracy	Precision	Recall	F-score
Porter stemmer	85.10	81.54	91.05	86.03
Lancaster stemmer	84.29	81.78	88.55	85.03
Lemmatization	85.09	81.72	90.69	85.96
Snowball stemmer	85.66	84.76	87.23	85.97

TABLE 11: Experiment 1: accuracy, precision, recall, and weighted F-score percentages by method for the CBOW model.

Method	Accuracy	Precision	Recall	F-score
Porter stemmer	85.90	85.99	86.03	86.01
Lancaster stemmer	85.67	86.37	84.98	85.67
Lemmatization	85.90	85.49	86.74	86.11
Snowball stemmer	85.52	86.03	85.08	85.55

TABLE 12: Experiment 2: accuracy, precision, recall, and weighted F-score percentages by method for the CBOW model.

Method	Accuracy	Precision	Recall	F-score
Porter stemmer	85.37	84.64	86.70	85.66
Lancaster stemmer	83.54	81.55	87.02	84.20
Lemmatization	85.50	83.16	89.30	86.12
Snowball stemmer	85.59	83.88	88.39	86.08

TABLE 13: Experiment 3: accuracy, precision, recall, and weighted F-score percentages by method for the CBOW model.

Method	Accuracy	Precision	Recall	F-score
Porter stemmer	85.26	87.30	83.30	84.99
Lancaster stemmer	85.31	83.06	89.01	85.93
Lemmatization	85.46	84.56	87.04	85.78
Snowball stemmer	85.02	82.02	90.00	85.83

TABLE 14: Experiment 4: accuracy, precision, recall, and weighted F-score percentages by method for the CBOW model.

Method	Accuracy	Precision	Recall	F-score
Porter stemmer	86.09	85.04	87.85	86.42
Lancaster stemmer	79.17	72.04	95.87	82.26
Lemmatization	86.66	85.67	88.29	86.96
Snowball stemmer	85.00	81.38	91.07	85.96

TABLE 15: Experiment 1: accuracy, precision, recall, and weighted F-score percentages by method for the Skip-gram model.

Method	Accuracy	Precision	Recall	F-score
Porter stemmer	85.50	86.47	84.44	85.44
Lancaster stemmer	82.68	89.40	74.46	81.25
Lemmatization	85.43	90.14	79.82	84.66
Snowball stemmer	85.92	84.81	87.78	86.27

TABLE 16: Experiment 2: accuracy, precision, recall, and weighted F-score percentages by method for the Skip-gram model.

Method	Accuracy	Precision	Recall	F-score
Porter stemmer	85.58	88.75	81.74	85.10
Lancaster stemmer	85.75	86.54	84.92	85.73
Lemmatization	86.18	85.37	87.58	86.46
Snowball stemmer	86.41	87.70	84.94	86.30

TABLE 17: Experiment 3: accuracy, precision, recall, and weighted F-score percentages by method for the Skip-gram model.

Method	Accuracy	Precision	Recall	F-score
Porter stemmer	86.27	85.06	88.25	86.63
Lancaster stemmer	80.74	74.31	94.42	83.17
Lemmatization	85.74	87.85	83.21	85.47
Snowball stemmer	85.75	84.12	88.41	86.21

TABLE 18: Experiment 4: accuracy, precision, recall, and weighted F-score percentages by method for the Skip-gram model.