Using Machine Learning to Predict the Sentiment of Online Reviews: A New Framework for Comparative Analysis

Gregorius Satia Budhi[a,b], Raymond Chiong[a], Ilung Pranata[a], and Zhongyi Hu[c]

[a] School of Electrical Engineering and Computing, The University of Newcastle, Callaghan, NSW 2308, Australia

[b] Informatics Department, Petra Christian University, Surabaya 60236, Indonesia

[c] School of Information Management, Wuhan University, Wuhan 430072, China

Abstract

Online reviews are becoming increasingly important for decision-making. Consumers often refer to online reviews for opinions before making a purchase; marketers acknowledge the importance of online reviews and use them to improve product success. However, the massive amount of online review data, as well as its unstructured nature, is a challenge for anyone wanting to derive a conclusion quickly. In this paper, we propose a novel framework for gauging the ratings of online reviews using machine learning techniques which uses a combination of text pre-processing and feature extraction methods. We investigate four different aspects of this new framework. First, we assess the performance of single and ensemble classifiers in predicting sentiment (initially positive or negative) on a specific dataset (Yelp), confirming they behave just as well when applied to different datasets. Second, using the best identified classifiers, we improve the accuracy with which neutral polarity can be predicted, an ability largely overlooked in the literature. Third, we further improve the performance of these classifiers by testing different pre-processing and feature extraction methods. Finally, we measure how well our deep learning approach performs on the same task compared to the best previously identified classifiers.

Our extensive testing with high powered computing shows that the linear kernel Support Vector Machine, Logistic Regression, and Multilayer Perceptron are probably the three-best single classifiers in terms of accuracy, precision, recall, and *F*-measure. Their performance could be further improved if they were used as base classifiers for ensemble models. We also observe that several text pre-processing techniques – negation word identification, word elongation correction, and part of speech lemmatisation (combined with Terms Frequency *N*-grams) – can increase accuracy. In addition, we demonstrate that the general sentiment of lexicons such as SentiWordNet3 and SenticNet 4 can be used to generate features with good results, although deep learning models can perform equally well.

Experiments with different datasets further confirm that our framework gives consistent outcomes, including with the Amazon and movie reviews datasets. We have focused on improving the accuracy of neutral sentiment, and we conclude by showing how this can be achieved without sacrificing the accuracy of positive or negative ratings.

Keywords: Online reviews and ratings, classification, machine learning, feature extraction, text mining.

1. INTRODUCTION

Online reviews of products and services play an important role for both buyers and sellers. On the one hand, consumers are paying more attention to the opinions of others on products they are interested in, in order to gauge a product's reliability and usefulness prior to making a purchase [1-3]. On the other hand, online reviews provide a tremendous wealth of feedback for marketers to understand the factors driving sales and trends, as well as to gauge the satisfaction level of consumers [4, 5]. The ability to find relevant content accurately and timely therefore helps both consumers and sellers make business decisions quickly [4, 6-8].

The rapid proliferation of web and social media sites provide various ways by which users can provide

reviews about a product or seller. This creates an unprecedented volume of data from which insights can be discovered [9], yet it is extremely challenging for anyone to read and assimilate it [10]. Thus, an automated system capable of analysing and finding relevant reviews easily and efficiently is of value in today's online environment [11, 12]. Automated review analysis involves training machines to capture and discriminate text polarity (positive or negative) from user reviews. The quality of this training process determines how accurately the review texts can be analysed, and how well they can be classified into rating categories [13].

The majority of existing studies, however, have only considered user opinions on products and/or services based on two polarities, namely positive and negative [6, 9, 14-30]. In other words, these studies overlook the middle ground or neutral polarity. Only a small number of studies have taken three polarities (or more) into account (e.g., see [3, 31, 32]). It has been shown here that ignoring neutral opinions incurs a loss of valuable information for decision making, and can even lead to wrong decisions [3, 33]. Omitting neutral opinion leads to either underestimating or overestimating negative or positive reviews [34]. The information contained within reviews with neutral polarity can impact product sales, and their existence can affect readers' perception about negative and positive reviews on products [34, 35]. Customers give a neutral rating because of an indifferent opinion about a product or service; the reviewer feels truly neutral or ambivalent since they find both positive and negative aspects of the product [33]. It also show that neutral opinions can help distinguish negative and positive ratings [36]. To reiterate, although neutral opinions are important, datasets that give three or more polarities are rare, with the majority just focusing on binary categories.

The above problems pose several questions that need to be answered. These are: (i) Is it possible to use raw, real-world datasets (such as the Yelp reviews dataset) to train classifiers for sentiment polarity detection, especially with more than two polarities? (ii) Which kind of classifiers, including deep learning models, are suitable for sentiment polarity detection? (iii) Can improvement be achieved by refining text pre-processing and feature extraction techniques? (iv) Can the best classifiers be applied to other datasets and provide similar results? To answer these questions, a comparison framework is established to examine various types of polarities, from 2-, 3-, to 5-polarity. Importantly, we pay special attention to neutral polarity detection. Several text pre-processing methods, such as negation word identification, word elongation correction, and part of speech (POS) lemmatisation, are applied. We also implement a number of Bag-of-Words (BoW) feature extraction methods, such as Terms Frequency (TF), Terms Frequency–Inverse Document Frequency (TF-IDF), and *N*-gram words. In addition to the BoW features, we investigate the option of using sentiment lexicons such as SentiWordNet 3.0 [37] and SenticNet 4 [38]. Varying sizes of input features and training samples are also considered to reduce the dimensionality of the review data. A range of single [39-45] and ensemble [46-50] classifiers, including Deep Learning (DL) models [51, 52], are evaluated using the following four metrics: accuracy, precision, recall, and *F*-measure. These classifiers are used to experiment on real word datasets obtained from Yelp [53], Amazon [54], and Large Movie Reviews (LMRs) [55]. The outcomes of these experiments may contribute to future research in this area and also be of value to online commerce websites where sentiment polarity prediction might be particularly useful.

The rest of this paper is organised as follows. In Section 2, we review work on polarity classification. In Section 3 we describe the methodology used, including system design, data, output targets, and classifiers. Section 4 discusses the experimental results, text pre-processing, classifiers, ensembles, deep learning, *N*-gram terms, and sentiment lexicons; it also sets out our investigations into neutral polarity. Finally, Section 5 concludes the work and highlights future research directions.

2. RELATED WORK

Studies related to polarity classification or prediction of online reviews have been on the rise in recent years. Previous work has used either product or service reviews. Some studies have focused on specific review datasets for more accurate and tuned results, while others have attempted to generalise their proposed methods by using wider datasets, including data from Twitter, debates, news, and of course product or service reviews. The majority of these studies have considered 2-polarity classification, using either manually labelled or unlabelled raw review datasets. Table 1 provides an overview of such studies.

Here, we first discuss studies based on labelled review data, and then look at unlabelled review data. Labelled

review datasets include Twitter's movie reviews, Cornel movie reviews, LMRs, restaurant reviews, and the like. These datasets have been manually labelled by experts based on sentiment polarity of the texts.

Basari et al. [14] used a Support Vector Machine (SVM) to detect the polarity of Twitter's movie reviews, with its parameters optimised by a particle swarm optimisation algorithm. Rong et al. [21] presented a method inspired by Bagging Predictors to recognise the negative or positive polarity of Cornel movie reviews and LMRs. Agarwal et al. [18] proposed a combination of ontology (ConceptNet), WordNet, and polarity lexicons (SenticNet, SentiWordNet, General Inquirer) to identify the polarity (negative/positive) of product and service reviews, including restaurant, movie, and software reviews manually labelled by experts. A framework for enhanced sentiment analysis and polarity classification (eSAP) was developed by Khan, Qamar, and Bashir [15]. This eSAP framework combines the SentiWordNet polarity lexicon and SVM to detect polarities of text and online movie reviews [15]. Khan et al. later extended their work and developed a sentiment dictionary named Senti-MI [16]. Tripathy et al. grouped text polarities into negative and positive using an *N*-gram model with classifiers such as Naïve Bayes and Maximum Entropy on IMDb movie review data [9]. Araque et al. attempted to improve the accuracy of deep learning by combining it with classical classifiers for predicting the 2-polarity of several text datasets, including IMDb movie reviews [19].

Besides manually labelled datasets, previous studies have also used raw datasets like Amazon's product reviews and TripAdvisor's hotel reviews for polarity detection. These studies relied on ratings given by users as the base of their 2-polarity prediction. Feature-based clustering was investigated by Bafna and Toshniwal to detect negative and positive polarities of Amazon's product reviews [20]. Fattah [23] extracted the polarities of Amazon's product reviews as well as Cornel movie reviews based on a new term-weighting scheme and a combination of some single classifiers. A method to evaluate the polarities of Amazon's cross-lingual reviews was proposed by Hajmohammadi et al. [24]. Around the same time, Katz, Ofek, and Shapira [17] proposed a context-based method named Consent which generates 3-gram key terms based on probability, and these key terms are used as features to detect polarity (negative/positive) using a Rotation Forest classifier. Katz et al. tested this method using manually labelled text datasets, including movie reviews, and also unlabelled raw data from TripAdvisor's hotel reviews [17]. Wang et al. [22] proposed a pipeline method based on a combination of random subspace for feature selection and an SVM-based ensemble of classifiers for text polarity classification. They tested their method on different review datasets, including movie reviews, Amazon's product reviews, and several service reviews [22]. Hung and Chen [25] proposed a word sense disambiguation technique to extract features from movie and hotel reviews, and built several classifiers to detect polarity (negative or positive). Ikram et al. [26] focused on detecting two polarities of Twitter's open source software products using classifiers such as AdaBoost and Apriori. Onan et al. proposed a multi-objective weighted voting ensemble classifier to classify sentiment polarities of online product and service reviews [27]. Vechtomova proposed several methods to detect negative and positive polarities of Amazon's product reviews at the word level without relying on training datasets and lexicons [28]. A combination of machine learning classifiers and sampling methods was proposed by Vinodhini and Chandrasekaran for 2-polarity sentiment classification of Amazon's product reviews having an unbalanced data distribution [30]. A feature extraction method, using document frequency, chi-square, information gain, standard deviations, and weighted log-likelihood ratios, was proposed by Yousefpour, Ibrahim, and Hamed [29] to classify the polarities of movie reviews and Amazon's product reviews.

While less common, several researchers have considered three or more polarities, which include neutral opinions, in their work. Fernández-Gavilanes et al. [32] proposed an unsupervised text classification method based on dependency parsing to classify texts from two (negative/positive) and three (negative/neutral/positive) polarities. They tested their proposed method using text datasets from Twitter and movie reviews [32]. Chen et al. [31] proposed a deep learning approach by combining Bidirectional Long Short-Term Memory with conditional random fields and a one-dimensional Convolution Neural Network (CNN). This approach works well for 2-polarity detection using data from movie reviews and Amazon's product reviews, but is less successful with 5-polarity prediction when tested with Stanford's sentiment treebank and its neutral sentiment [31]. An intuitional fuzzy-weighted averaging operator and preference-ranking organisation methods were developed by Liu et al. for 3-polarity detection (positive/neutral/negative) using automobile reviews [3]. Budhi et al. [56] investigated the use of supervised machine learning methods for 2- and 3-polarity prediction on Yelp 2017 review data. A system named SHAN (Syntax-directed Hybrid Attention Network) was built using a combination of several Bi-LSTM to detect the three polarities of sentiment in text (negative, neutral, and positive) [57]. López et al. [58] proposed E^2SAM (Evolutionary Ensemble of Sentiment Analysis Methods), which is a set of sentiment analysis methods to detect 3-polarity sentiment in texts.

Table 1. An overview of related work

Two polarities		**Three or more polarities**	**Polarity detection as part of other application**
Manually classified datasets	**Raw datasets**		
Basari et al. [14]	Bafna & Toshniwal [20]	Gavilanes et al. [32]	Bagheri, Saraee & de Jong [11]
Rong et al. [21]	Fattah [23]	Chen, et al. [31]	Hur, Kang & Cho [59]
Agarwal, et al. [18]	Hajmohammadi et al. [24]	Liu, Bi & Fan [3]	Zhang et al. [60]
Fattah [23]	Katz, Ofek & Shapira [17]	Budhi, et al. [56]	Gui et al. [61]
Katz, Ofek & Shapira [17]	Wang et al. [22]	Wang et al. [57]	Zhang et al. [62]
Wang et al. [22]	Hung & Chen [25]	López et al. [58]	
Hung & Chen [25]	Ikram et al. [26]		
Khan, Qamar & Bashir [6, 15, 16]	Khan, Qamar & Bashir [6, 15, 16]		
Tripathy et al. [9]	Onan et al. [27]		
Araque, et al. [19]	Vechtomova [28]		
Yousefpour, Ibrahim & Hamed [29]	Vinodhini & Chandrasekaran [30]		
	Yousefpour, Ibrahim & Hamed [29]		

Other researchers have used text polarity detection as part of their applications. For example, Bagheri, Saraee, and de Jong proposed an unsupervised model using heuristic rules for an iterative bootstrapping algorithm and aspect pruning. They used this method to extract and detect explicit and implicit aspects of Amazon's product reviews [11]. Text mining of movie reviews and factors such as nationalities, ratings, and other qualitative variables were considered by Hur et al. [59] for box-office forecasting based on a Korean movie review dataset. Zhang et al. [60] used sentiment orientation as one of the verbal features to detect fake reviews from the Yelp dataset using verbal and non-verbal features. Gui et al. [61] proposed a method to classify product reviews based on heterogeneous network representations, which included users (opinion holders), words, products (opinion targets), and polarities (positive and negative). They processed these network representations using different classifiers, and found that CNNs had the best results for the datasets tested, which included IMDb movie reviews, Yelp 2013, and Yelp 2014 [61]. Zhang et al. [62] proposed MOCA – Multi-Objective, Collaborative, and Attentive sentiment analysis – to predict the overall ratings of texts such as customer reviews from IMDb and Yelp 2013 and Yelp 2014.

Our work differs from all these studies in that its comparisons are made by considering a number of well-known machine learning models, including both single and ensemble classifiers and other classifiers from the deep learning family. In terms of datasets, we have used the unlabelled Yelp 2017 and Amazon product reviews, as well as the labelled Large Movie Review (LMR) dataset. Our focus is on improving the accuracy of 3-polarity classification, given that only a few related studies have considered it and that the results to date are far from satisfactory.

3. METHODS

To evaluate the use of machine learning techniques to successfully rate polarity from review texts, we propose the comparison framework shown in Fig. 1. Here, the loaded reviews are first processed by removing punctuation, numbers, and common words. Features are then extracted from the texts using bag-of-words combined with TF or TF-IDF. Targets related to polarities are extracted based on different settings. Single and ensemble machine learning techniques are applied to build the prediction models. Finally, comparisons of different models are made based on four metrics, and statistical tests are used to gauge the differences between them.

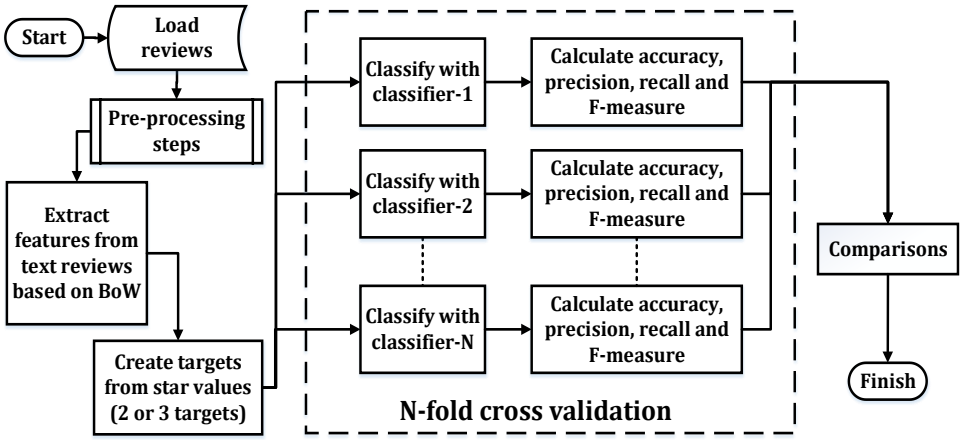

Fig. 1. A comparison framework for rating polarity

3.1. Experimental Data and Labels

Consumer review data from the Yelp Dataset Challenge Round 9 in 2017 is the primary dataset used in our study. Yelp is a leader in consumer ratings, and has grown rapidly since 2005. Yelp's users can review local businesses like restaurants, hair salons, bars, pubs, and many others. Users write their reviews and give star ratings from 1 to 5 to any business listed with Yelp [63]. The dataset used in this work contains 4.1 million review texts. Processing and experimenting on a massive dataset is a big challenge, and we relied on the high-performance computing facilities at the University of Newcastle, Australia. We used distributed servers having a total of 2560 cores, 66 CPUs, and 4 GPU nodes, where each node could be assigned up to 256 GB RAM.

To predict the polarity of Yelp 2017 review data, we made use of the 1–5 star ratings given on each review as our target label. Based on the star ratings, three main experimental output target types were created by categorising the review texts as follows.
Type A: negative reviews were reviews with 1-star and 2-star ratings, while positive reviews were those with 3-, 4-, and 5-star ratings;
Type B: negative reviews were reviews with 1-, 2-, and 3-star ratings, while positive reviews were those with 4- and 5-star ratings;
Type C: negative reviews were reviews with 1- and 2-star ratings, neutral reviews were those with 3-star ratings only, and positive reviews were those with 4- and 5-star ratings.
Later, for more detailed analysis we created another five types of experimental output targets (see 4.2 below).

Other review datasets used in our experiments included Amazon's product reviews and LMRs. The Amazon dataset [54] is a large dataset containing more than 100 million product reviews. It is much larger in comparison with the Yelp 2017 dataset. Like the Yelp 2017 review data, the dataset is unlabelled, and products are ranked from 1–5. For our experiments, we made use of the 1–5 ratings given on each review as our target label, and three output targets were considered: Types A and B for 2-polarity classification, and Type C for 3-polarity classification. The LMR dataset [55] is a prepared and manually tagged dataset for research purposes. It is quite different to Yelp 2017 and Amazon's product reviews in that it has 50,000 records where each record is manually labelled as either a positive or negative review. For this dataset we therefore have to restrict the target output to just two polarities.

3.2. Pre-processing Steps

Prior to generating features for machine learning, we have to pre-process the review texts. Fig. 2 shows the pre-processing steps we have applied to the three review datasets. These pre-processing steps involve removal of punctuation, numbers, and English stop words, tokenisation of words, and token lemmatisation. We used NLTK

modules [64] to clean the review texts of punctuation and numbers as well as tokenise and lemmatise each word. Our lemmatisation approach has three steps: first, each word is lemmatised as a noun; then a verb; and finally an adverb. This is to reduce the words to their basic form. Subsequently, the words are joined based on their original order and saved. To improve the results, we later implement negative word processing, word elongation correction, and POS lemmatisation.

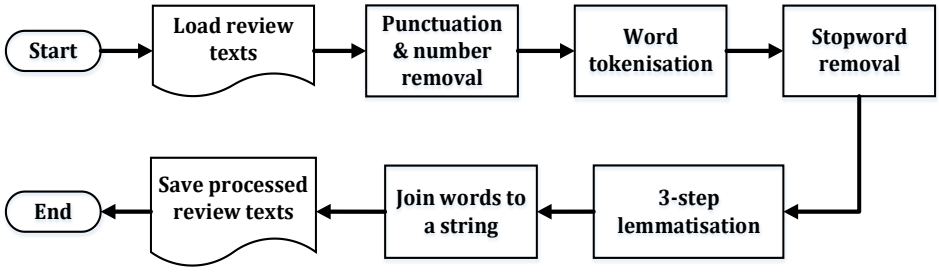

Fig. 2. Pre-processing steps

3.3. Feature Extraction

Feature extraction determines how features are selected and is important in influencing the accuracy of automated review analysis [65]. After pre-processing, we extract machine learning features from the processed review texts. We used Terms Frequency (TF) to generate features for each pre-processed word token. The process to create features is as follows. First, a bag of words (BoW) from all samples is created and then their TF values are calculated. Next, they are sorted based on their TF values. Features for each review text are extracted by checking for the existence of each feature word. If a feature word does not exist in the review text, then 0 is assigned. Otherwise, the feature word's TF value is calculated and assigned to the matrix of features. A feature set can be created from all unique words found in the review texts, or a subset of them above a certain threshold value. To increase prediction accuracy, we later replace TF with TF-IDF.

The Yelp 2017 review dataset has more than 4.1 million review records. The total number of unique words in this dataset after pre-processing is more than 240K. A problem arises when all unique words are used as features. This creates more than 984 billion values and so requires a huge memory allocation for model training, even with our high-performance grid computers. To find manageable sizes of features and samples, we performed experiments with various settings to reduce the number of features and samples used in training.

3.4. Classifiers

In our work we considered not only standard single classifiers but also ensemble models, and compared their performances against each other. In total, 13 single and 5 ensemble classifiers commonly used for classification and text mining tasks were examined. In the following, we first describe the single classifiers, followed by the ensemble models. All classifiers used in this study were built using the Scikit-Learn module for machine learning and the Keras module for deep learning which are commonly used for these purposes [66-68].

3.4.1 Single models

Naïve Bayes is often used in classification problems [69, 70], including text classification [60, 71, 72]. It is the simplest form of Bayesian network classifiers if each feature is independent. Many applications have successfully

implemented Naïve Bayes, and it is considered to be one of the top 10 data mining algorithms [73]. In this study, we investigated three types of Naïve Bayes classifiers: *Multinomial* and *Bernoulli Naïve Bayes* [39] and *Gaussian Naïve Bayes* [74].

The idea of Nearest Neighbour classifiers is to cluster instances into groups based on their closest distance [40]. First introduced by Fix and Hodges in 1951 [75], Nearest Neighbour classifiers are widely used in different studies [72, 76-79]. In this work, we investigated two types of Nearest Neighbour classifiers, namely the *K-Nearest Neighbour* [40] and *Nearest Centroid* [80].

The Generalised Linear Model (GLM) was firstly proposed by Nelder and Wedderburn in 1972 [81], and subsequently improved by Hastie and Tibshirani in 1990 [82]. It is a generalisation of the linear regression model and attempts to overcome several limitations that the former has. The GLM was developed with non-normal dependent variables [83, 84]. There are many variants of this model, and they have been used to solve a wide range of classifications problems [85-89]. In this work, we investigated four types of GLM: *Logistic Regression* [41], *Ridge Regression* [90], *Passive Aggressive* [42], and *Stochastic Gradient Descent* [91].

The *Decision Tree* classifier was developed by Quinlan [92] based on Hunt's algorithm [93]. As the name suggests, it is a tree-like model, creating decision trees for classification and prediction purposes. The classifier is a useful explanatory tool for expressing a cause-and-effect chain [43]. It has been used for text classification [94, 95] as well as many other applications [96, 97]. This algorithm is typically used as a base classifier for ensemble methods (e.g., Bagging, Random Forest, and AdaBoost).

The SVM learns from a training dataset and generalises to make correct predictions on unseen data. It works by separating a hyperplane into classes and then maximising the separation distance. The larger the margin, the lower the error generated by the classifier [44]. The excellent generalisation performance of SVM makes it very popular in many research areas [72, 98-103]. In this study, we investigated SVMs with Linear and (Gaussian) Radial Basis Function kernels [104].

The *Multilayer Perceptron* is a feedforward Artificial Neural Network normally used as a supervised model for pattern recognition and classification [59]. The model works by minimising error through computing weights in its network. The algorithm continually updates the weights to achieve the best configuration. It consists of two phases, feed-forward and backpropagation. In the feed-forward phase, training data is forwarded to produce an output, and then the difference between the real output and desired target is calculated to produce an error. This error is then used to update the weights [45]. The algorithm has been used and improved by many researchers in different areas [72, 105-110].

3.4.2 Ensemble models

Bagging Predictors use several single predictors to build a cluster of predictors. The predictors are trained through a bootstrapping process that replicates the training set. Bagging uses plurality votes to predict a class [46] and is commonly used in many areas [27, 72, 111, 112]. In this study, we investigated Bagging Predictors with different single classifiers, including the Decision Tree, Logistic Regression, Linear-Kernel SVM, and Multilayer Perceptron.

The *Random Forest* is an ensemble of decision tree predictors, in which each tree is trained using a random vector that is sampled independently. Error generalisation of Random Forest depends on the strength of each individual tree and the correlation between them. This ensemble model is relatively robust to outliers and noise [47], and is used in many areas including text classification [4, 72, 89, 113]. In addition to the standard Random Forest, in this study we also investigated the *Randomised Decision Trees*, another variant of decision tree ensemble classifiers [49].

AdaBoost is short for Adaptive Boosting. This algorithm iteratively combines multiple weak classifiers over several rounds, starting with equal weights for all training data. If training data points are misclassified, their weights are boosted, and then a new classifier is created using the new unequal weights. This process is repeated for the set of

classifiers [50]. AdaBoost has been successfully used for identifying malicious web domains, predicting financial distress dynamically, speaker verification, and imbalanced data classification [72, 100, 114, 115].

Gradient Boosting is an ensemble of gradient-boosted regression trees for classifying dirty data. It produces a robust competitive and interpretable algorithm for classification and regression. However, it uses only a single regression tree for binary classification [48]. This algorithm has been applied to many classification and regression problems [72, 116, 117].

3.4.3 Deep learning

The term 'deep' in Deep Learning (DL) refers to the concept of numerous abstract layers created when data is transformed or converted from input to output [118]. DL techniques offer the capability of learning features in both supervised and unsupervised ways. DL architectures are mainly based on Artificial Neural Networks (ANNs) with multiple hidden layers between the input and output. They have been shown to learn features accurately [119]. In our experiments, we implemented several types of DL for detecting sentiment polarity as explained in the remainder of this section.

Convolutional neural network (CNN) has been successfully used in pattern recognition, computer vision, and sentiment analysis [31, 52, 120-122]. In general, CNN consists of convolutional layers that create features for the network to learn. These convolution layers can be complemented with normalisation layers and pooling layers. Normally, the convolution layers are flattened with fully connected layers and followed by a softmax layer for performing classification or pattern recognition [52, 120, 122]. By varying the number of layers and nodes/neurons in each layer, CNN has fewer connections and parameters and is easy to train. Theoretically, its training performance is slightly worse than the standard feedforward neural network [120].

Long short-term memory (LSTM) was proposed by Hochreiter and Schmidhuber in 1997 [51]. It is a special type of Recurrent Neural Network and is capable of learning long term dependencies. Modules in LSTM include four interacting layers: input, output, cell state, and forget gate. Every memory cell contains a node with a fixed weight of 1 and with a self-connected recurrent edge to avoid gradients vanishing or exploding [123].

4. EXPERIMENTAL RESULTS AND DISCUSSIONS

4.1. Experiments on Classifiers

These experiments were intended to investigate what are the best single classifiers, and also the best ensembles to use for gauging sentiment polarity. In these experiments, we investigated several well known classifiers and ensembles. We mainly used the Yelp review dataset for our initial experiments. However, we also used other datasets, such as Amazon reviews and movie reviews, to test whether each classifier's performance was consistent across the datasets.

4.1.1 Experiments on single classifiers

We investigated the performance of single classifiers in identifying review polarity, first using the Yelp 2017 dataset. A total of 13 classifiers, as shown in Table 2, were tested using three types of experiment (i.e., Types A, B, and C) as defined in section 3.1.

We performed 10-fold cross-validation based on 10,000 randomly selected review texts using each of the 13 classifiers. In these experiments, we ran the classifiers with varying numbers of features ranging from 250 to the maximum (245,071). The features were selected based on their TF values. The accuracies of the classifiers with

different feature sets in the three experiment types are shown in Fig. 3.

As shown in Fig. 3, classifiers such as BNB, GNB, DT, KNN, NC, and RSVM did not perform well compared to the others. MNB can perform well with limited features, but the accuracy deteriorates when the number of features increases. The results also show that the accuracy of all classifiers, except RR, does not increase any further when the number of features is beyond 5,000. RR reaches its peak accuracy at around 10,000 features. These results indicate that increasing the number of features, which increases training complexity, does not necessarily increase the accuracy of the training models.

Table 2. Single classifiers and their settings

No.	Classifier Name	Parameter
1	Multinomial Naïve Bayes (MNB)	alpha = 1.0
2	Bernoulli Naïve Bayes (BNB)	alpha = 1.0
3	Gaussian Naïve Bayes (GNB)	–
4	K-Nearest Neighbour (KNN)	K = 5, Euclidean
5	Nearest Centroid (NC)	Euclidean
6	Decision Tree (DT)	Gini index
7	Logistic Regression (LR)	max iterations: 100
8	Ridge Regression (RR)	alpha = 1.0
9	Passive Aggressive (PA)	Epochs = 5, PA-I formula
10	Stochastic Gradient Descent (SGD)	estim: Linear SVM, learning rate = 1.0 / (alpha * ($t + t_0$))
11	RBF-kernel SVM (RSVM)	gamma = 1/*n* features
12	Linear-kernel SVM (LSVM)	max iterations = 1000
13	Multilayer Perceptron (MLP)	1 hidden layer - 100 neurons, rectified linear unit, $\alpha = 0.001$

Next, we performed cross-validation using 500 features sorted by their TF values, on various review texts ranging from 10,000 records to the maximum of 4,133,088 records (i.e., the entire Yelp review dataset). Fig. 4 shows the accuracies of these classifiers with an increasing number of records on the three experiment types. From Fig. 4, we see that the accuracies of these classifiers increase quite substantially as the number of records increases until it reaches about 500K. Beyond this point, the increase in accuracy is marginal.

From Fig. 3 and Fig. 4, we can see that the MLP, LR, and LSVM are the best performers in most cases. Additionally, we observe that experiment Type A has the highest accuracy, followed by Types B and C.

To test the robustness of the three best classifiers, we performed further experiments with varying features (i.e., 1,000 to 10K feature sets) and training samples (i.e., 100K to 1M training samples) based on 10-fold cross-validation. The results for each of the classifiers are shown in Table 3.

Table 3. Results of the three best single classifiers on Yelp 2017 review data

Classifier Name	Experiment Type	Training Time (*)	Maximum Accuracy (%)	Average (%)			
				Accuracy	Precision	Recall	*F*-measure
LSVM	A	0.07	90.57	89.26	89.13	89.26	89.14
	B	0.08	86.94	85.40	85.34	85.40	85.32
	C	0.10	82.25	80.83	79.09	80.83	79.37
LR	A	0.06	90.90	90.17	89.94	90.17	90.00
	B	0.05	87.23	86.46	86.34	86.46	86.37
	C	0.06	82.82	82.04	79.92	82.04	80.41
MLP	A	1.84	91.23	90.38	90.33	90.38	90.35
	B	0.87	87.54	86.12	86.13	86.12	86.12
	C	0.68	82.65	81.47	80.82	81.47	81.09

(*) Average training time per sample training record in millisecond

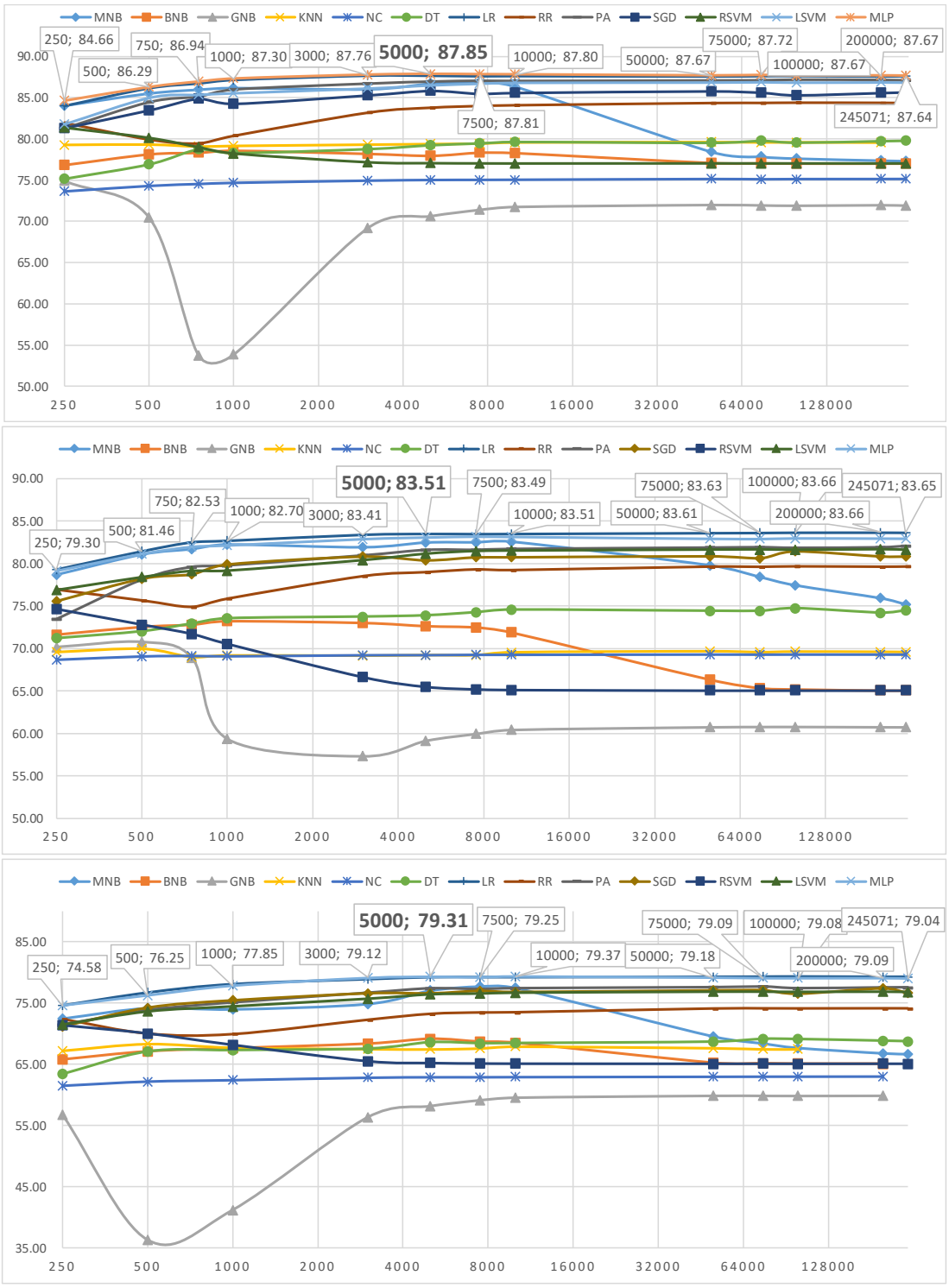

Fig. 3. Accuracy (*y*-axis) vs the number of features (*x*-axis) for single classifiers based on the Yelp 2017 review dataset. Top, Experiment type A; middle, type B; bottom, type C

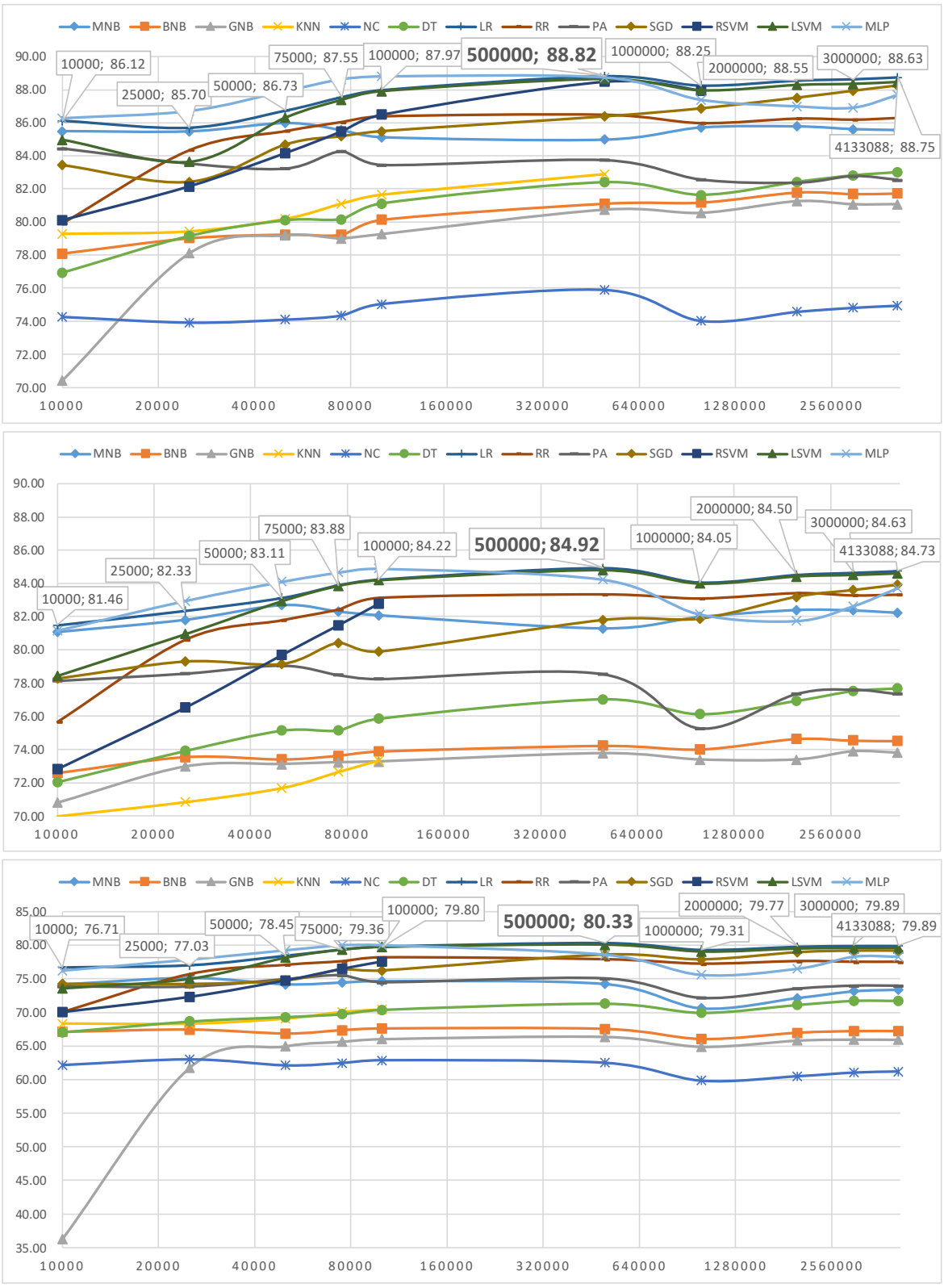

Fig. 4. Accuracy (*y*-axis) vs the number of training records (*x*-axis) for single classifiers based on the Yelp 2017 review dataset. Experiment Type A (top); B (middle); and C (bottom)

From the results, we observe that the combination of optimal features and the optimal amount of training data increases the accuracy, precision, recall, and *F*-measure of the three best classifiers. MLP has best results for Type A and LR for Type B. As for the Type C target output, the best accuracy and recall results are obtained by LR. The MLP has the best results in terms of precision and *F*-measure. Although MLP is the best classifier for Type A experiments, it required the longest training time. Having said that, the training time of MLP on average is only around one millisecond per record, which is acceptable.

4.1.2 Experiments on ensemble classifiers

Besides single classifiers, we also investigated the performance of five ensemble classifiers and their variants as listed in Table 4. We used Yelp 2017 dataset for these investigations. Results of these experiments can be seen in Table 5. The experiments were conducted in the same manner as those in Table 3.

Table 4. Ensemble classifiers and their settings

No.	Classifier Name	Parameter
1	Randomized Decision Trees	10 estimators (DT), Gini index
2	Random Forest	10 estimators (DT), Gini index
3	Gradient Boosting	loss function: LR, 100 estimators (LR), mean squared error
4	Bagging Predictors	10 estimators (DT), bootstrap: true
5	Bagging (LSVM)	10 estimators (LSVM), bootstrap: true
6	Bagging (LR)	10 estimators (LR), bootstrap: true
7	Bagging (MLP)	10 estimators (MLP), bootstrap: true
8	AdaBoost	50 estimators (DT)
9	AdaBoost (LSVM)	50 estimators (LSVM)
10	AdaBoost (LR)	50 estimators (LR)

Table 5. Results of the ensemble classifiers on Yelp 2017 review data

Classifier Name	Experiment Type	Training Time (*)	Maximum Accuracy (%)	Average (%)			
				Accuracy	Precision	Recall	*F*-measure
Rand. Decision Trees	A	0.06	80.43	79.64	80.13	79.64	79.86
	B	0.05	74.57	73.74	74.08	73.74	73.89
	C	0.05	69.19	68.04	68.20	68.04	68.09
Gradient Boosting	A	5.38	86.98	86.64	86.42	86.64	85.15
	B	4.53	82.57	82.28	82.46	82.28	81.39
	C	9.50	78.52	78.23	75.85	78.23	74.55
Random Forest	A	0.11	87.95	87.19	86.73	87.19	86.86
	B	0.06	83.04	82.20	82.36	82.20	82.26
	C	0.09	79.27	78.52	76.48	78.52	76.19
Bagging	A	2.12	87.15	86.38	86.38	86.38	86.37
	B	1.66	82.16	81.35	81.71	81.35	81.48
	C	2.47	78.29	77.45	75.62	77.45	76.05
AdaBoost	A	0.62	87.26	86.75	86.07	86.75	85.99
	B	1.06	82.79	82.31	82.02	82.31	81.94
	C	0.70	77.88	77.59	74.30	77.59	74.55
Bagging (LSVM)	A	0.50	90.58	89.26	89.14	89.26	89.14
	B	0.32	86.96	85.40	85.34	85.40	85.33
	C	1.10	82.26	80.83	79.10	80.83	79.37
AdaBoost (LSVM)	A	0.15	90.57	89.26	89.14	89.26	89.14
	B	0.11	86.95	85.40	85.34	85.40	85.33
	C	0.38	82.25	80.83	79.10	80.83	79.36
Bagging (LR)	A	0.70	91.16	90.36	90.12	90.36	90.18
	B	0.23	87.54	86.65	86.53	86.65	86.55
	C	0.64	83.19	82.27	80.11	82.27	80.60
AdaBoost (LR)	A	0.49	89.51	88.78	88.80	88.78	88.78
	B	0.58	85.69	84.78	84.81	84.78	84.79
	C	0.79	80.56	79.44	78.49	79.44	78.89
Bagging (MLP)	A	7.18	92.11	91.21	91.08	91.21	91.13
	B	6.59	88.81	87.47	87.42	87.47	87.44
	C	8.01	84.27	83.39	82.00	83.39	82.45

(*) Average training time per sample training record in millisecond.

By default, both Bagging and AdaBoost have Decision Tree as their base classifier. However, it is possible to change the base classifier. In this study, we further investigated the performance of Bagging and AdaBoost by replacing their base classifier with each of the aforementioned three best single classifiers. However, we did not use MLP as the base classifier for AdaBoost since it can only combine classifiers that support sample weighting.

By comparing results in Table 3 and the first 15 rows of Table 5, we observe that, on all metrics, the three best single classifiers in Table 3 are better than all these five ensemble classifiers. However, as can be read in sub section 4.1.1., the performance of DT, which is the default base classifier for these five ensemble models, is not as good as those classifiers listed in Table 3. This might explain why the ensemble models' results are the worst of all those listed in Table 3. Looking at the second part of Table 5 (on the results for Bagging and AdaBoost with the three best single classifiers as base classifiers), we note that the performance of Bagging is improved but not for AdaBoost. Direct comparisons for the three best classifiers as stand-alone classifiers vs. as base classifiers can be found in Table 6.

Table 6. Results of using the best single classifiers as base classifiers for the ensemble models

Classifier Name	Experiment Type	Average Accuracy (%) (*)		
		Single	In Bagging	In AdaBoost
LSVM	A	89.26	89.26	89.26
	B	85.40	85.40	85.40
	C	80.83	80.83	80.83
LR	A	90.17	90.36 ↑	88.78 ↓
	B	86.46	86.65 ↑	84.78 ↓
	C	82.04	82.27 ↑	79.44 ↓
MLP	A	90.38	91.21 ↑	-
	B	86.12	87.47 ↑	-
	C	81.47	83.39 ↑	-

(*) ↑ / ↓ = increase/decrease

4.1.3 Verification

For verification purposes, we conducted non-parametric statistical analysis based on a Wilcoxon signed-rank test to see if results between the three best single classifiers and ensemble models are significantly different. Due to space constraints, we present only the statistical test results based on *F*-measure. These results can be seen in Table 7, Table **8**, and Table ***9***. In these tables, *p*-values that are greater than the significance level (i.e., > 0.05) are highlighted in bold. Almost all the pairwise comparisons are significantly different except for LSVM, Bagging (LSVM), and AdaBoost (LSVM). We can conclude that using LSVM as the base classifier for these ensemble models is less useful.

Table 7. Wilcoxon signed-rank test results for each pair of classifiers on Type A experiments (*F*-measure)

	Linear SVM	MLP	RDT	GB	RF	Bagging (DT)	Bagging (LR)	Bagging (LSVM)	Baging (MLP)	AdaBoost (DT)	AdaBoost (LR)	AdaBoost (LSVM)
LR	7.85E-15	8.44E-08	7.85E-15	7.85E-15	7.85E-15	7.85E-15	1.51E-13	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15
Linear SVM		7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	0.016803	7.85E-15	7.85E-15	9.09E-05	**0.521972**
MLP			7.85E-15	7.85E-15	7.85E-15	7.85E-15	0.011641	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15
RDT				7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15
GB					8.47E-15	1.26E-13	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15
RF						7.48E-13	7.85E-15	7.85E-15	7.85E-15	2.51E-14	7.85E-15	7.85E-15
Bagging (DT)							7.85E-15	7.85E-15	7.85E-15	3.57E-08	7.85E-15	7.85E-15
Bagging (LR)								7.85E-15	8.15E-15	7.85E-15	7.85E-15	7.85E-15
Bagging (LSVM)									7.85E-15	7.85E-15	7.29E-05	**0.288914**
Bagging (MLP)										7.85E-15	7.85E-15	7.85E-15
AdaBoost (DT)											7.85E-15	7.85E-15
AdaBoost (LR)												5.84E-05

Table 8. *Wilcoxon signed-rank test results for each pair of classifiers on Type B experiments (F-measure)*

	Linear SVM	MLP	RDT	GB	RF	Bagging (DT)	Bagging (LR)	Bagging (LSVM)	Bagging (MLP)	AdaBoost (DT)	AdaBoost (LR)	AdaBoost (LSVM)
LR	7.85E-15	**0.080837**	7.85E-15	7.85E-15	7.85E-15	7.85E-15	2.43E-10	7.85E-15	1.09E-13	7.85E-15	7.85E-15	7.85E-15
Linear SVM		5.33E-11	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	0.021456	7.85E-15	7.85E-15	4.43E-06	**0.628082**
MLP			7.85E-15	7.85E-15	7.85E-15	7.85E-15	0.003599	6.68E-11	7.85E-15	7.85E-15	6.31E-14	4.84E-11
RDT				7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15
GB					1.56E-08	**0.68703**	7.85E-15	7.85E-15	7.85E-15	8.79E-15	7.85E-15	7.85E-15
RF						8.79E-15	7.85E-15	7.85E-15	7.85E-15	0.002249	7.85E-15	7.85E-15
Bagging (DT)							7.85E-15	7.85E-15	7.85E-15	2.48E-06	7.85E-15	7.85E-15
Bagging (LR)								7.85E-15	2.45E-11	7.85E-15	7.85E-15	7.85E-15
Bagging (LSVM)									7.85E-15	7.85E-15	4.33E-06	0.035601
Bagging (MLP)										7.85E-15	7.85E-15	7.85E-15
AdaBoost (DT)											7.85E-15	7.85E-15
AdaBoost (LR)												5.20E-06

Table 9. Wilcoxon signed-rank test results for each pair of classifiers on Type C experiments (*F*-measure)

	Linear SVM	MLP	RDT	GB	RF	Bagging (DT)	Bagging (LR)	Bagging (LSVM)	Bagging (MLP)	AdaBoost (DT)	AdaBoost (LR)	AdaBoost (LSVM)
LR	7.85E-15	2.45E-11	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15
Linear SVM		7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	**0.088628**	7.85E-15	7.85E-15	4.18E-13	**0.988302**
MLP			7.85E-15	7.85E-15	7.85E-15	7.85E-15	1.36E-08	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15
RDT				7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15
GB					7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	0.007551	7.85E-15	7.85E-15
RF						0.00349	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15
Bagging (DT)							7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15
Bagging (LR)								7.85E-15	7.85E-15	7.85E-15	7.85E-15	7.85E-15
Bagging (LSVM)									7.85E-15	7.85E-15	9.89E-13	0.040558
Bagging (MLP)										7.85E-15	7.85E-15	7.85E-15
AdaBoost (DT)											7.85E-15	7.85E-15
AdaBoost (LR)												1.55E-12

4.1.4 Predicting with other datasets

To investigate whether our comparison framework performs well on other datasets, we performed similar experiments with two additional datasets: Amazon's product reviews [54] and LMR [55, 124].

The Amazon dataset [54] contains more than 100 million product reviews. In our experiments, we used only a subset of its review dataset, particularly reviews on clothes, shoes, and jewellery products. This subset has about 5 million records, which is of similar size to Yelp 2017 dataset. We conducted experiments in a similar fashion to those in Table 3 and Table **5**. Classification results can be found in Table 10. Similar to Yelp results, MLP has the best accuracy, precision, recall, and *F*-measure scores compared to LSVM and LR. However, their differences are marginal. When using these three best single classifiers as the base classifiers in Bagging, their accuracy increases. Similar to Yelp case, the Type A experiments has higher accuracy compared to Type B where the 3-star rating is negative. We can conclude that, in Amazon dataset, users who gave 3-star rating tend to provide more positive comments than negative ones. Type C has the worst results but the scores are still higher than 80%. 3-polarity detection is normally more difficult than 2-polarity detection.

Table 10. Results based on the Amazon review dataset

Classifier Name	Experiment Type	Training Time(*)	Maximum Accuracy (%)	Average (%)			
				Accuracy	Precision	Recall	*F*-measure
LSVM	A	0.08	91.15	90.12	89.49	90.12	89.60
	B	0.20	86.87	85.65	85.16	85.65	85.21
	C	0.11	83.80	82.33	79.57	82.33	80.20
LR	A	0.09	91.41	90.89	90.12	90.89	90.26
	B	0.12	87.58	86.57	86.00	86.57	86.07
	C	0.08	83.65	83.21	80.06	83.21	80.83
MLP	A	1.77	91.67	91.02	90.59	91.02	90.75
	B	2.92	87.88	86.57	86.30	86.57	86.40
	C	1.89	83.66	82.65	81.48	82.65	81.99
Bagging (LSVM)	A	0.43	91.12	90.19	89.58	90.19	89.69
	B	0.62	86.92	85.68	85.21	85.68	85.27
	C	0.58	83.81	82.27	79.52	82.27	80.06
Bagging (LR)	A	0.40	91.48	90.96	90.18	90.96	90.29
	B	0.42	87.61	86.70	86.14	86.70	86.20
	C	0.44	84.09	83.24	80.05	83.24	80.79
Bagging (MLP)	A	15.39	92.14	91.39	90.78	91.39	90.93
	B	23.22	88.75	87.16	86.73	87.16	86.84

C	23.81	85.76	84.27	82.25	84.27	82.89

(*) Average training time per sample training record in the millisecond.

So far our investigation has focused on the raw datasets, i.e. Yelp 2017 and Amazon. To investigate whether our approach can be applied to a manually prepared dataset, we conducted experiments using the LMR dataset. Recall that, compared to Yelp 2017, the LMR dataset [55] is small, but it has been prepared and designed for research purposes. We conducted 10-fold cross-validation using all of its records (50,000) based on the previously obtained best single classifiers (Table 3) and Bagging ensembles (Table 5). We varied the number of features from 250 to a maximum of 44,346. These features were selected based on their TF values. The accuracy of these classifiers and with different features are shown in **Error! Reference source not found.**. For a direct comparison, we also ran experiments using 50,000 records selected randomly from the Yelp 2017 dataset with features varying from 250 to 50,000, as shown in **Error! Reference source not found.**.

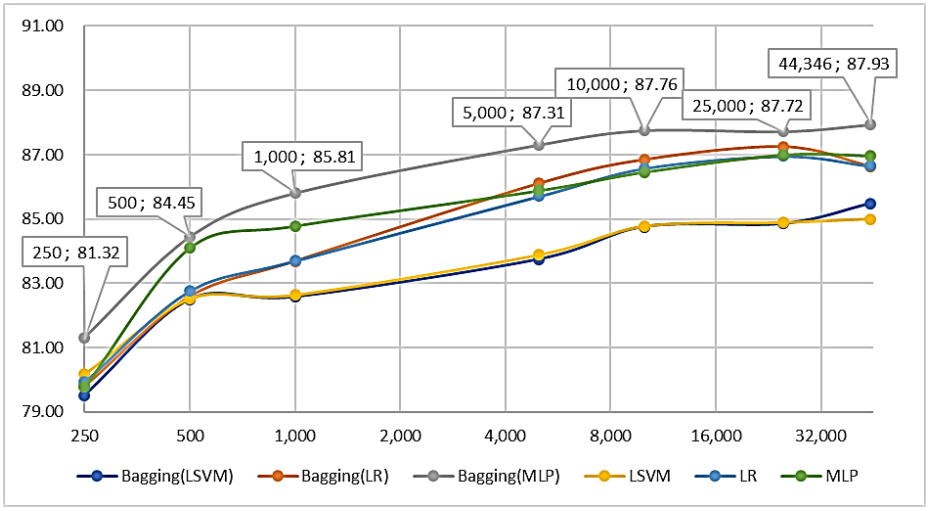

Fig. 5. Accuracy (*y*-axis) vs the number of features (*x*-axis) for different classifiers based on the LMR dataset

From **Error! Reference source not found.**, we can see that the performance of LSVM for the LMR dataset is the worst among all three single classifiers. Having this classifier in Bagging does not increase the accuracy either. Meanwhile, the best accuracy is acquired by Bagging with MLP. We also observe that the increase in accuracy is marginal after 5,000 features. This observation is consistent with results obtained from the Yelp 2017 dataset.

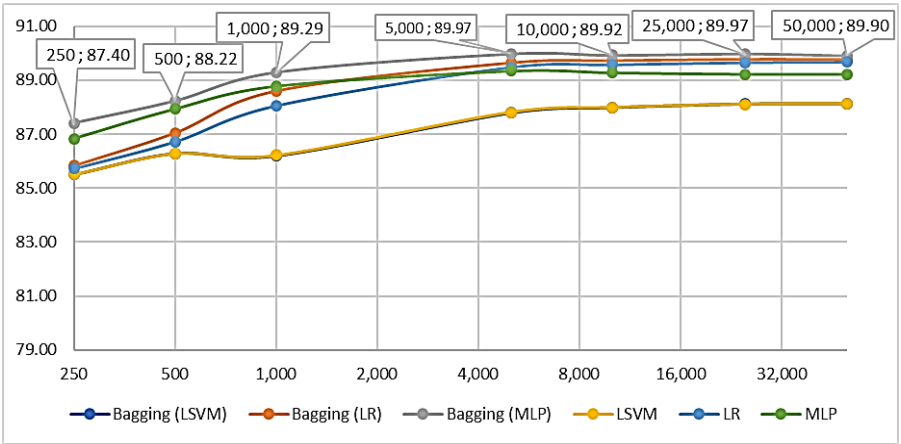

Fig. 6. Accuracy (*y*-axis) vs the number of features (*x*-axis) for different classifiers based on the Yelp 2017 dataset

4.2. Experiments on Neutral Polarity

In this experiment, we focus on detecting neutral polarity. Here we have added five more experimental types as set out below.

a. Type D: negative reviews are reviews with 1-star rating; neutral reviews are those with 2- and 3-star ratings; and positive reviews are those with 4- or 5-star ratings.

b. Type E: negative reviews are reviews with 1- and 2-star ratings; neutral reviews are those with 3- and 4-star ratings; and positive reviews are those with a 5-star rating.

c. Type F: negative reviews are reviews with a 1-star rating; neutral reviews are those with 2-, 3-, and 4-star ratings; and positive reviews are those with a 5-star rating.

d. Type G: Split star ratings into five sentiment polarities.

e. Type H: negative reviews are reviews with a 1-star rating; neutral reviews are those with a 3-star rating; and positive reviews are those with a 5-star rating. That is, we exclude 2- and 4-star reviews from the experiments.

These experiments were conducted using Bagging Predictors with the three best single classifiers as base classifiers. We used the Yelp 2017 dataset for these experiments with the same classifier parameters as in previous experiments. The result are set out in **Error! Reference source not found.**.

Error! Reference source not found. shows that a neutral rating (3 stars) for Type C has the lowest accuracy. Type C is 1 or 2 stars for negative polarity, 3 stars for neutral, and 4 or 5 stars as positive. This indicates that the trained classifiers are not quite capable of recognising 'neutral' reviews. Predicting neutral ratings is always more challenging because neutral reviews may not have an equal make up of positive and negative comments. Most of the time, neutral rating comments tend to skew towards one target class (positive or negative). This can be seen from the results of Types A and B, where we assumed the 3-star rating to be either negative or positive.

As can be seen in Type D experiments, when the 2- and 3-star ratings are considered as neutral, the accuracy of 3-star rating increases significantly compared to Type C experiments where only 3-star reviews are considered neutral. However, the classification accuracy for 1- and 2-star then decreases. This is because many of the 1-star reviews are very similar to 2-star reviews and thus they are misclassified as neutral. Similarly, when 3- and 4-star reviews are assumed to be neutral (Type E), the accuracy of 3-star classification increases. These results strengthen the findings from previous experiments (with Types A, B, and C) which suggest that users who give a 3-star rating tend to provide more positive comments than negative ones.

From the experiment we know that middle-rating stars (2 and 4) can increase accuracy when added to the

extreme ratings, i.e. to 1 or 5 stars or the neutral rating (3 stars). In C, D, E, and F types, we found that 2 stars can contribute to negative or neutral with similar effect, while the 4 star is more useful when applied to positive polarity. From these, we can conclude that people who give 2 stars can have negative or neutral opinions, while the majority of reviewers think that 4 stars is positive.

In Type F experiments, we see that the accuracy of neutral polarity is significantly increased when we consider 2- and 4-star reviews as neutral. However, the accuracy for both positive and negative polarities then decreases. The reason for this is that 1- and 5-star reviews normally have similar features (review texts) to 2- and 4-star reviews respectively. Users who provide 5-star ratings normally write similar reviews to those who provide 4-star ratings. This phenomenon also appears for those who provide 1- and 2-star ratings.

Table 11. Accuracy comparison between target types based on the Yelp 2017 review dataset

Classifier Name	Experiment Type	Average Accuracy (%)	Average accuracy of targets (%)			Average accuracy of stars (%)				
			Negative	Neutral	Positive	1	2	3	4	5
Bagging (LSVM)	A (12-345)	89.26	72.45	-	93.96	81.78	57.78	79.84	95.31	97.26
	B (123-45)	85.40	76.56	-	89.94	89.21	82.44	58.51	82.63	94.23
	C (12-3-45)	80.83	76.65	25.90	92.30	84.81	63.79	25.90	86.60	95.64
	D (1-23-45)	79.87	68.72	48.82	91.81	68.72	51.88	46.68	85.35	95.59
	E (12-34-5)	70.74	75.86	60.99	76.60	85.32	61.00	65.84	58.60	76.60
	F (1-234-5)	71.80	65.87	70.44	75.12	65.87	70.40	83.06	64.25	75.12
	G (1-2-3-4-5)	61.33	-	-	-	74.11	38.21	40.63	48.14	75.85
	H (1-3-5)	79.33	84.41	61.06	92.51	84.41	-	61.06	-	92.51
Bagging (LR)	A (12-345)	90.36	73.40	-	95.09	83.39	57.69	81.12	96.65	98.24
	B (123-45)	86.65	77.52	-	91.33	90.84	83.85	58.37	84.02	95.63
	C (12-3-45)	82.27	78.16	25.35	94.05	86.86	64.48	25.35	88.76	97.15
	D (1-23-45)	81.33	69.35	50.30	93.41	69.35	54.16	47.59	87.06	97.14
	E (12-34-5)	72.39	76.82	63.59	77.76	87.08	60.70	68.98	60.94	77.76
	F (1-234-5)	73.74	67.26	73.39	76.16	67.26	73.74	86.77	66.66	76.16
	G (1-2-3-4-5)	61.30	-	-	-	72.48	27.76	30.62	45.81	82.38
	H (1-3-5)	80.69	85.53	63.34	93.19	85.53	-	63.34	-	93.19
Bagging (MLP)	A (12-345)	91.21	77.32	-	95.09	86.96	62.14	81.79	96.59	98.09
	B (123-45)	87.47	80.43	-	91.08	92.89	86.05	62.72	84.01	95.24
	C (12-3-45)	83.39	81.02	34.67	93.08	89.51	67.65	34.67	87.37	96.43
	D (1-23-45)	82.28	73.19	55.98	92.24	73.19	59.09	53.79	85.66	96.07
	E (12-34-5)	73.35	78.79	66.70	76.25	88.17	63.93	70.65	64.75	76.25
	F (1-234-5)	74.69	72.07	74.72	75.46	72.07	73.28	86.52	69.40	75.46
	G (1-2-3-4-5)	61.28	-	-	-	72.19	29.23	34.39	47.63	79.97
	H (1-3-5)	82.59	87.27	67.71	92.78	87.27	-	67.71	-	92.78

Type G experiments confirm that the extreme ratings, i.e. 1 and 5 stars, can be easily identified from the reviews as these are on the extremely disappointed or satisfied scale. We can see that the accuracy for other stars, i.e. 2, 3, and 4 stars, suffers much. The drop of accuracy is because the opinions of users in between these stars are similar. This is an important finding, as the distinction between 2, 3, and 4 stars does not matter much as they are all considered neutral. The experiments of Type G also prove that predicting sentiment polarities to more than 3 polarities using the raw dataset are difficult since the review texts of the neutral polarities tend to look similar to each other. When the 2- and 4-star reviews are excluded in Type H, the classifier's accuracy for other stars (1, 3, and 5 stars) improves significantly. This is because without 2- and 4-star reviews, reviews for each rating class are very distinctive and hence easier to classify.

4.3. Techniques to Improve Existing Experimental Results

Several improvements can be made to the previous experiments. In this section, we explain these refinements, mainly in text pre-processing and feature-extraction techniques, which improve the outcomes.

4.3.1 Experiment on BoW TF-IDF featuring

In the previous experiments, we used BoW TF to extract features from the review texts. To improve the experimental outcomes, we now investigate the BoW TF-IDF technique to extract features. We conduct the experiments in the same manner as in Table 3 and Table 5 and use the same dataset. By comparing the results in Table 12 with those in Table 3 and Table 5, we observe that models based on TF-IDF features have similar performance accuracy (about 1% difference) compared to TF-based models**Error! Reference source not found.**. Furthermore, for most classifiers in the experiments the training time is not significantly different between the TF and TF-IDF methods. With the TF-IDF feature extraction technique, the training time of Bagging (LSVM) and Bagging (LR) decreases, although the decrease is insignificant. In terms of training performance, we observe that Bagging (MLP) needs more training time.

Table 12. Results for TF-IDF featuring based on the Yelp 2017 review dataset

Classifier Name	Experiment Type	Training Time (*)	Maximum Accuracy (%)	Average (%)			
				Accuracy	Precision	Recall	*F*-measure
Bagging (LSVM)	A	0.13	90.66	90.25	90.04	90.25	90.11
	B	0.15	86.70	86.29	86.20	86.29	86.23
	C	0.21	82.90	82.14	79.89	82.14	80.14
Bagging (LR)	A	0.28	90.44	89.98	89.71	89.98	89.57
	B	0.19	86.85	86.45	86.32	86.45	86.29
	C	0.43	82.27	81.83	79.39	81.83	78.83
Bagging (MLP)	A	17.89	91.45	90.56	90.41	90.56	90.47
	B	24.80	88.06	86.57	86.53	86.57	86.54
	C	26.10	83.81	82.59	81.18	82.59	81.65

(*) Average training time per sample training record in millisecond.

Table 13. Comparison of training time and average accuracy between TF and TF-IDF featuring

Classifier Name	Experiment Type	Training Time (*)		Average Accuracy (%)	
		TF	TF-IDF	TF	TF-IDF
Bagging (LSVM)	A	0.50	0.13↓	89.26	90.25↑
	B	0.32	0.15↓	85.40	86.29↑
	C	1.10	0.21↓	80.83	82.14↑
Bagging (LR)	A	0.70	0.28↓	90.36	89.98↓
	B	0.23	0.19↓	86.65	86.45↓
	C	0.64	0.43↓	82.27	81.83↓
Bagging (MLP)	A	7.18	17.89↑	91.21	90.56↓
	B	6.59	24.80↑	87.47	86.57↓
	C	8.01	26.10↑	83.39	82.59↓

(*) Average training time per sample training record in millisecond;
↑ / ↓ = increase/decrease

4.3.2 Experiments on negation, word elongation, and part of speech lemmatisation

Negation words can affect the polarity of a whole sentence [125]. Here we process negation words such as “don’t”, “doesn’t”, “shouldn’t”, etc., back to their basic words such as “do not”, “does not”, ‘should not’, etc. By returning them to their basic form, ‘not’, we reduce the diversity of the texts. Word elongation or word-stretchers such as “Yesss”, “Fiiine”, “Yoouu”, etc. add to the subjectivity value of a sentence since people who do so are trying

to show emotion in the text [126]. Word elongation can also increase the diversity of the words in data training, and make the classifiers harder to train. We therefore correct them back to their basic words using Peter Norvig's code for spelling correction [127]. The code is based on probability theory in which the chosen word is compared to words from a large text source and the most likely candidate is chosen as the replacement. We use this same correction method to correct "n't" to "not" after separating it from its basic word, e.g. "don't" → "do n't".

We also replaced 3-step lemmatisation using Part of Speech (POS) lemmatisation. The 3-step lemmatisation that we previously used forced all words to be one of their basic forms, disregarding the POS type of the words. Back then, we thought it would make the word more general and reduce diversification. However, POS tagging is implemented in some sentiment polarisation researches, since some parts of speech express polarity [125]. To implement POS lemmatisation, we first tag the POS type of the words, and after that the words are lemmatised to their basic forms based on their POS tags. For tagging and lemmatising, we used a component from NLTK [64]. The design of our new pre-processing can be seen in Fig. 7. The results of the additional pre-processing are set out in **Table 14** (the BoW-Unigram column). When we compare these results to those of the previous pre-processing version (**Error! Reference source not found.Error! Reference source not found.**), the additional pre-processing step can increase accuracy and other measurements from 0.5% to 1% or more. However, when we look in more detail, there is no increase in accuracy and other measurements for Type H. In Type H experiments we used a special sub-dataset in which we isolated the negative (1-star), neutral (3-star), and positive (5-star) ratings by deleting 2- and 4-star reviews.

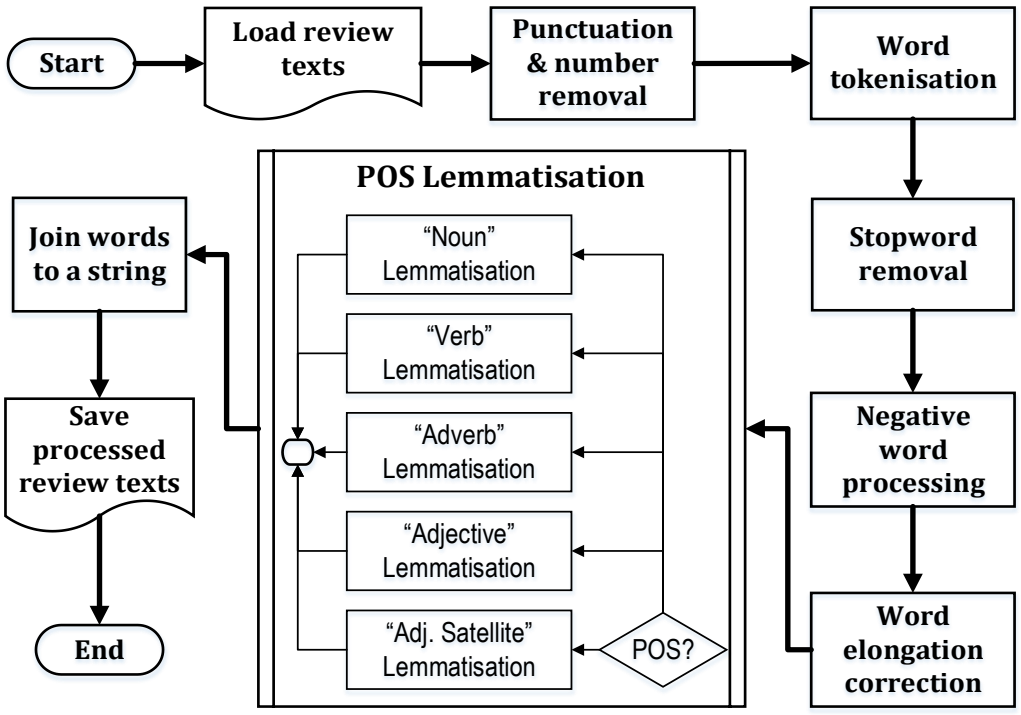

Fig. 7. Pre-processing step with negation word, word elongation, and POS lemmatisation

4.3.3 Experiments on *N*-gram Bag of Words and sentiment lexicons

In this section, we explore the possibility of using *N*-gram words as features. Compared to the unigram words from the previous experiments, here we use bigram and trigram words. We stop the *N*-gram experiments with trigrams because we could not find 4-gram word features when we processed the dataset for feature extraction. The results of these experiments are in **Table 14**, where we also show results of using sentiment lexicons as features. Here we used two sentiment lexicons, SentiWordNet 3.0 [37] and SenticNet 4 [38]. Sentiment lexicon is a bag of sentiment words which were prepared by experts and given sentiment scores. The scores are given positive/neutral/ negative values in SentiWordNet 3.0, and positive/negative values in SenticNet 4.

Table 14. Accuracy of Bag of Words and two sentiment lexicons (%)

Polarity Type	Classifier	Bag of Words			SentiWordNet3 (bigram)		SenticNet 4 (bigram)	
		Unigram	Bigram	Trigram	w. score	w/o score	w. score	w/o score
Type A	Bagging (MLP)	91.75	92.03	92.05	91.12	91.70	91.48	91.52
(12-345)	Bagging (LR)	90.92	91.34	91.23	90.23	91.01	90.60	90.92
	Bagging (LSVM)	89.76	90.49	90.29	90.28	90.05	89.93	89.90
Type B	Bagging (MLP)	88.26	88.69	88.77	88.10	88.02	87.63	87.57
(123-45)	Bagging (LR)	87.34	87.78	87.67	87.07	87.22	86.42	86.67
	Bagging (LSVM)	86.49	87.10	86.97	86.31	86.41	85.90	85.90
2-Polarity Avg. Accuracy		**89.09**	**89.57**	**89.50**	**88.85**	**89.07**	**88.66**	**88.75**
Type C	Bagging (MLP)	84.06	84.57	84.57	82.97	83.83	83.45	83.33
(12-3-45)	Bagging (LR)	82.93	83.49	83.37	81.94	82.90	82.10	82.45
	Bagging (LSVM)	81.86	82.81	82.64	81.95	82.04	81.49	81.56
Type H	Bagging (MLP)	82.58	82.95	83.23	80.23	82.08	80.82	80.88
(1-3-5)	Bagging (LR)	80.70	81.51	81.37	77.34	80.35	78.41	79.04
	Bagging (LSVM)	79.33	80.28	80.09	77.63	79.14	77.57	77.86
3-Polarity Avg. Accuracy		**81.91**	**82.60**	**82.55**	**80.34**	**81.72**	**80.64**	**80.85**
Overall Avg. Accuracy		**85.50**	**86.09↑**	**86.02↑**	**84.60↓**	**85.40↓**	**84.65↓**	**84.80↓**

From these experiments we can see that including two words or multiple words (bigram or trigram) as the features has a positive effect compared to single words (unigram), but it is not large (only 0.5%). This small effect is because the number of bigram words in the features is not significant compared to the unigram, and even less compared to the trigram. The number of trigrams in the features is so small so that they do not increase the accuracy, and sometimes they even have a slightly negative effect.

Instead of the bag of words, we also investigated the effect of using sentiment lexicon as the features. Sentiment lexicon is the group of special words with a connection to opinions or emotions. Some researchers have made an effort to build sentiment lexicons in order to use it for detecting sentiment in a text. There is an advantage to using a sentiment lexicon compared to a bag of words. In the bag of words model, we need to create a bag of words feature every time we want to train the classifier. It means the features we produce depend on a certain set of data. On the contrary, a sentiment lexicon is built with the intention of using it for general purpose sentiment analysis, so they are independent of the data set. Usually, a sentiment lexicon is created along with a set of fixed procedures. However, instead of using such procedures, here we investigated the performance of two well-known sentiment lexicons, SentiWordNet 3.0 and SenticNet 4, when used to train classifiers, and we have used this process to predict the sentiment of customer reviews.

The experiments were done using a sentiment lexicon with a unigram, bigram, and trigram approach. There are 4-gram words (and more) in both sentiment lexicons; however, 4-gram words do not exist in the dataset we used. The results were consistent with BoW, where the unigram had slightly lower accuracy compared to the bigram, while the trigram was similar to the bigram. For comparison with the BoW, Table 14 shows the accuracy of bigram SentiWordNet 3.0 and SenticNet 4. On average, the results of the experiments using bigram sentiment lexicons were slightly worse than using the unigram BoW, especially when we included the sentiment score given by the authors as features. This is because when the scores are included, the features become more diverse. However, the small difference shows that the effect of sentiment scores in the features is not significant.

In Table 14, the accuracy of bigram sentiment lexicons is lower than unigram BoW. When we investigated this in detail, for 2-polarity (Type A and Type B) the differences were small (0.2% to 0.4%), while for 3-polarity (Type C and Type H) the difference was more than 1% for SenticNet 4 (without score) and less than 0.2% for SentiWordNet 3.0. This shows that SenticNet 4 is less suitable for detecting 3-polarity than SentiWordNet 3.0. This is because SenticNet 4 consists only of positive and negative words. Furthermore, similar results were obtained when we compared the result of Type A to the results published in the original SenticNet 4 paper [38]. We therefore conclude that the SenticNet 4 lexicon can be used as features for training machine learning classifiers for sentiment analysis, giving the same result as the author's method. The authors of SentiWordNet 3.0 did not test the method they proposed, so we could not do a comparison. However, based on the good results of our experiments in Table 13, we can conclude that SentiWordNet 3.0 can also be used as a feature base to train classifiers for sentiment analysis.

4.4 Predicting using Deep Learning

Because of their good results, Deep Learning (DL) algorithms are a machine learning tool that is now frequently used by researchers as classifiers or predictors. We have investigated several types of common DL tools such as Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Feed Forward DL (FFDL). FFDL is a multilayer perceptron which is implemented using a deep learning library such as Theano or Keras [128]. Here we used Keras to build DL models [67]. First, we ran small experiments using 10,000 records from the Yelp Reviews dataset based on 1000 features, two targets (Type 2), unigram, and 10-fold cross-validation. For the experiments, we built five models of one layer FFDL. We implemented the default setting of MLP in Table 2 to create the FFDL-Base. We also built three LSTM models (one of them was a combination of CNN and LSTM), two CNN models that we based on the Simple CNN model [129], and a Very Deep CNN model [52, 130]. The configuration of the DL models we used and the results are shown in Table 15, Table 16, and Table 17.

Table 15. FFDL models, 10,000 records, 1000 features, Experiment Type A

DL model	Configuration	Average (%)			
		accuracy	precision	recall	F1
FFDL Base	Dense(100, relu) - Dense(2, softmax)	87.43	87.41	87.43	87.40
FFDL 1	Dense(1000, relu) - Dense(2, softmax)	87.72	87.65	87.72	87.66
FFDL 2	Dense(6000, relu) - Dense(2, softmax)	87.81	87.74	87.81	87.75
FFDL 3	Dense(12000, relu) - Dense(2, softmax)	88.04	87.97	88.04	87.97
FFDL 4	Dense(18000, relu) - Dense(2, softmax)	88.09	88.02	88.09	88.02

Table 16. LSTM models, 10,000 records, 1000 features, Experiment Type A

DL model	Configuration	Average (%)			
		accuracy	precision	recall	F1
LSTM 1	LSTM(100) - Dense(2, softmax)	66.72	61.42	66.72	56.16
LSTM 2	LSTM(200) - Dense(2, softmax)	67.18	52.88	67.18	54.95
CNN-LSTM	2 x Convolution(128, relu, kernel 3x3) - MaxPooling - LSTM(100) – Dense(2, softmax)	68.00	65.73	68.01	60.00

Table 17. CNN models, 10,000 records, 1000 features, Experiment Type A

DL model	Configuration	Average (%)			
		accuracy	precision	recall	F1
CNN model 1	2 x Convolution(32, relu, kernel 3x3) - MaxPooling - 2 x Convolution(64, relu, kernel 3x3) - MaxPooling - Dense(512, relu) - Dense(2, softmax)	83.31	83.16	83.31	83.13
CNN model 2	2 x Convolution(32, relu, kernel 3x3) - MaxPooling - 2 x Convolution(64, relu, kernel 3x3) - MaxPooling - 2 x Convolution(128, relu, kernel 3x3) - MaxPooling - Dense(1024, relu) - Dense(512, relu) - Dense(2, softmax)	82.69	82.70	82.69	82.66
Very Deep CNN [52, 130]	2 x Convolution(64, relu, kernel 3x3) - MaxPooling - 2 x Convolution(128, relu, kernel 3x3) - MaxPooling - 3 x Convolution(256, relu, kernel 3x3) - MaxPooling - 3 x Convolution(512, relu, kernel 3x3) - MaxPooling - 3 x Convolution(512, relu, kernel 3x3) - MaxPooling - 2 x Dense(4096, relu) - Dense(2, softmax)	57.16	34.95	57.16	42.87

From the experiments using small data (10,000 records), several interesting facts emerged. In FFDL (**Table 15**) we can see that adding a neuron to the processing (hidden) layer increases the accuracy and other measurements, but the increase is minimal (0.3%). After reaching a particular point, the accuracy plateaus. Moreover, LSTM is not a

good choice for answering the problem and it achieved low accuracy (**Table 16**). Although adding more nodes to it can increase accuracy, in general LSTM performs much more poorly compared to other methods. We also conducted experiments with a combination of CNN and LSTM in which we added two convolution layers to learning and added more features before continuing to the LSTM layer. The result was better, since the addition of 2 convolution layers increased accuracy by more than 1% and also greatly increased precision and other measurements, although they were still lower than with other methods. This is reasonable, since while LSTM is usually good at predicting serial data such as time series, the nature of the data featuring that we applied is not serial but simply reflects the existence of words in the data and disregards their order.

After the CNN-LSTM experiment, we took another approach: implementing Convolution layers which directly forwards the features from convolution layers to the predicted Dense layers (Table 17). For relatively small data sets, our CNN models performed well, although not better than the FFDL models. The more convolution layers we added, the worse the accuracy, and when we applied the Very Deep CNN setting the accuracy and other measurements fell below 50%. Our analysis differs from image processing in that the analysis of text reviews doesn't need deep feature extraction. In text processing the text needs to be pre-processed, either with Word2Vec [131], Doc2Vec [132], or other techniques, to reformat the input to a fixed form. In our research, however, we used a combination of BoW and TF, so in terms of text processing, the input of DL is already in the form of features, and doesn't need very deep featuring.

Finally, taking the models which showed good results, we conducted experiments in the same Big Data setting as used in previous experiments (500,000 records, 5000 features, unigram, Type A). The aim was to compare DL models with the best classifier found so far (Table 3 and Table 5, experiment Type A). The results of these experiments are set out in Table 18.

Table 18. Deep Learning models, 500,000 records, 5000 features, Type A

DL model	Configuration	Average (%)			
		accuracy	precision	recall	F1
FFDL Base	Dense(100, relu) - Dense(2, softmax)	90.62	90.58	90.62	90.60
CNN model 1	2 x Convolution(32, relu, kernel 3x3) - MaxPooling - 2 x Convolution(64, relu, kernel 3x3) - MaxPooling - Dense(512, relu) - Dense(2, softmax)	90.28	90.12	90.28	90.17
CNN model 2	2 x Convolution(32, relu, kernel 3x3) - MaxPooling - 2 x Convolution(64, relu, kernel 3x3) - MaxPooling - 2 x Convolution(128, relu, kernel 3x3) - MaxPooling - Dense(1024, relu) - Dense(512, relu) – Dense(2, softmax)	91.30	91.10	91.30	91.13

It is well known that DL performs better with Big Data. Therefore, we applied FFDL-Base, CNN model 1, and CNN model 2 to large-scale review data. Table 18 confirms that the FFDL-Base models give a similar result to MLP in previous experiments. On the other hand, CNN performed well in big data, so that the CNN model 1 that we built (based on a simple CNN model [129]) could reach an accuracy and other measurements similar to MLP. CNN model 2 achieved a better result, one that was similar to the Bagging (MLP). From these experiments, we conclude that DL, especially CNN, can be effectively used for sentiment polarity prediction since the accuracies achieved are similar to the best machine learning that we previously found (Table 3 and Table 5).

5. CONCLUSION AND FUTURE WORK

Ratings and reviews are important in order that potential customers can make purchase decisions and that sellers can obtain feedback on their products. To classify the massive amount of reviews into different polarities, this study has proposed a comparison framework which makes use of various machine learning and feature extraction techniques. Using the framework, comparison experiments were carried out using three real-world review datasets: the Yelp 2017 review data, Amazon product reviews, and LMRs. We investigated several feature extraction methods including TF or TF-IDF in a BoW approach, *N*-gram terms, and sentiment lexicons.

From these experiments we conclude that having more features or data in the training set does not necessarily

improve model performance. After reaching a certain threshold, the model performance plateaus. Our experiments indicate that 5,000 features and 500,000 reviews are the cut-off points for polarity prediction. The use of negation, correcting for word elongation, and POS lemmatisation can increase accuracy by 1%, while *N*-gram word feature extraction can increase accuracy slightly further. However, the *N*-gram method is limited to bigram words since words longer than bigram are rare in real-world datasets. Our experiments using sentiment lexicons for feature extraction show that accuracy (and associated measurements) are slightly lower than unigram BoW. But the success of sentiment lexicons for feature extraction is noteworthy, since sentiment lexicons are independent of the sample data whereas the BoW technique depends strongly on the sample.

We identified three single classifiers – the LR, LSVM, and MLP – which had better performance compared to others. They obtained accuracy, precision, recall, and *F*-measure scores above 90% or more for Type A experiments, above 87% for Type B experiments, and above 82% for Type C experiments in both the Yelp 2017 and Amazon review datasets. Further improvements could be achieved by utilising these three classifiers as the base classifiers in ensemble models. The implementation of DL, especially CNN for sentiment prediction, is possible since their models achieve similar measurements scores with ML classifiers and the ensemble. For a smaller dataset like LMR, the highest accuracy is slightly lower (87%). However, the similar results when comparing a manually polarised dataset (LMR) and a raw real-world dataset (Yelp Reviews 2017) convinces us that it is possible to use the stars or rankings given by the writers of these reviews as the basis of accurately gauging sentiment polarity.

We found that classifying neutral ratings (3 stars) is more challenging due to the fact that neutral reviews do not tend to have an equal distribution of positive and negative comments. In fact, we noticed that users who gave 3-star ratings had a tendency to give more positive reviews. Further experiments on use of three polarities, i.e., Types D and E, strengthened the finding that users giving a 3-star rating tend to give more positive reviews. The Type G experiments, with results for 5-polarity detection, show that creating a classifying system of more than 3-polarity is quite difficult, since the contents of some middle opinions (2, 3, and 4 stars) are vague and quite similar to each other. In our Type H experiments, we found that the accuracy of neutral polarity can be increased further if the in-between stars (2 and 4 stars) are removed. Three polarities can be predicted quite well, with an average overall accuracy of more than 85% and neutral polarity accuracy of more than 60%.

In future work, we plan to explore advanced feature selection techniques and bio-inspired optimisation algorithms which might help to improve the performance of the machine learning models considered in this study, especially in classifying neutral reviews.

ACKNOWLEDGEMENT

The first author would like to acknowledge financial support from the Indonesian Endowment Fund for Education (LPDP), Ministry of Finance, and the Directorate General of Higher Education (DIKTI), Ministry of Research, Technology and Higher Education, The Republic of Indonesia.

REFERENCES

[1] Z. P. Fan, Y. J. Che, and Z. Y. Chen, "Product sales forecasting using online reviews and historical sales data: A method combining the Bass model and sentiment analysis," *Journal of Business Research*, vol. 74, pp. 90-100, 2017.

[2] A. Y. K. Chua and S. Banerjee, "Helpfulness of user-generated reviews as a function of review sentiment, product type and information quality," *Computers in Human Behavior*, vol. 54, pp. 547-554, 2016.

[3] Y. Liu, J. W. Bi, and Z. P. Fan, "Ranking products through online reviews: A method based on sentiment analysis technique and intuitionistic fuzzy set theory," *Information Fusion*, vol. 36, pp. 149-161, 2017.

[4] A. Felbermayr and A. Nanopoulos, "The role of emotions for the perceived usefulness in online customer reviews," *Journal of Interactive Marketing*, vol. 36, pp. 60-76, 2016.

[5] Y. Ma, G. Chen, and Q. Wei, "Finding users preferences from large-scale online reviews for personalized recommendation," *Electronic Commerce Research*, vol. 17, no. 1, pp. 3-29, 2017.

[6] F. H. Khan, U. Qamar, and S. Bashir, "SWIMS: Semi-supervised subjective feature weighting and intelligent model selection for sentiment analysis," *Knowledge-Based Systems*, vol. 100, pp. 97-111, 2016.

[7] N. Jing, T. Jiang, J. Du, and V. Sugumaran, "Personalized recommendation based on customer preference mining and sentiment assessment from a Chinese e-commerce website," *Electronic Commerce Research*, vol. 18, no. 1, pp. 159-179, 2018.
[8] H. Zhang, H. Rao, and J. Feng, "Product innovation based on online review data mining: a case study of Huawei phones," *Electronic Commerce Research*, vol. 18, no. 1, pp. 3-22, 2018.
[9] A. Tripathy, A. Agrawal, and S. K. Rath, "Classification of sentiment reviews using n-gram machine learning approach," *Expert Systems with Applications*, vol. 57, pp. 117-126, 2016.
[10] M. Salehan and D. J. Kim, "Predicting the performance of online consumer reviews: A sentiment mining approach to big data analytics," *Decision Support Systems 81*, pp. 30-40, 2016.
[11] A. Bagheri, M. Saraee, and F. de Jong, "Care more about customers: Unsupervised domain-independent aspect detection for sentiment analysis of customer reviews," *Knowledge-Based Systems*, vol. 52, pp. 201-213, 2013.
[12] E. Fersini, E. Messina, and F. A. Pozzi, "Expressive signals in social media languages to improve polarity detection," *Information Processing & Management*, vol. 52, no. 1, pp. 20-35, 2016.
[13] M. D. Devika, C. Sunitha, and G. Amal, "Sentiment analysis: A comparative study on different approaches," *Procedia Computer Science*, vol. 87, pp. 44-49, 2016.
[14] A. S. H. Basari, B. Hussin, I. G. P. Ananta, and J. Zeniarja, "Opinion mining of movie review using hybrid method of Support Vector Machine and Particle Swarm Optimization," *Procedia Engineering*, vol. 53, pp. 453-462, 2013.
[15] F. H. Khan, U. Qamar, and S. Bashir, "eSAP: A decision support framework for enhanced sentiment analysis and polarity classification," *Information Sciences*, vol. 367-368, pp. 862-873, 2016.
[16] F. H. Khan, U. Qamar, and S. Bashir, "SentiMI: Introducing point-wise mutual information with SentiWordNet to improve sentiment polarity detection," *Applied Soft Computing*, vol. 39, pp. 140-153, 2016.
[17] G. Katz, N. Ofek, and B. Shapira, "ConSent: Context-based sentiment analysis," *Knowledge-Based Systems*, vol. 84, pp. 162-178, 2015.
[18] B. Agarwal, N. Mittal, P. Bansal, and S. Garg, "Sentiment analysis using common-sense and context information," *Computational Intelligence & Neuroscience*, Article vol. 2015, pp. 1-9, 2015.
[19] O. Araque, I. Corcuera-Platas, J. F. Sánchez-Rada, and C. A. Iglesias, "Enhancing deep learning sentiment analysis with ensemble techniques in social applications," *Expert Systems with Applications*, vol. 77, pp. 236-246, 2017.
[20] K. Bafna and D. Toshniwal, "Feature based summarization of customer's reviews of online products," *Procedia Computer Science*, vol. 22, pp. 142-151, 2013.
[21] W. Rong, Y. Nie, Y. Ouyang, B. Peng, and Z. Xiong, "Auto-encoder based bagging architecture for sentiment analysis," *Journal of Visual Languages & Computing*, vol. 25, no. 6, pp. 840-849, 2014/12/01/ 2014.
[22] G. Wang, Z. Zhang, J. Sun, S. Yang, and C. A. Larson, "POS-RS: A Random Subspace method for sentiment classification based on part-of-speech analysis," *Information Processing & Management*, vol. 51, no. 4, pp. 458-479, 2015.
[23] M. Abdel Fattah, "New term weighting schemes with combination of multiple classifiers for sentiment analysis," *Neurocomputing*, vol. 167, pp. 434-442, 2015.
[24] M. S. Hajmohammadi, R. Ibrahim, A. Selamat, and H. Fujita, "Combination of active learning and self-training for cross-lingual sentiment classification with density analysis of unlabelled samples," *Information Sciences*, vol. 317, pp. 67-77, 2015.
[25] C. Hung and S. J. Chen, "Word sense disambiguation based sentiment lexicons for sentiment classification," *Knowledge-Based Systems*, vol. 110, pp. 224-232, 2016.
[26] M. T. Ikram, N. A. Butt, and M. T. Afzal, "Open source software adoption evaluation through feature level sentiment analysis using Twitter data," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 24, pp. 4481-4496, 2016.
[27] A. Onan, S. Korukoğlu, and H. Bulut, "A multiobjective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification," *Expert Systems with Applications*, vol. 62, pp. 1-16, 2016.
[28] O. Vechtomova, "Disambiguating context-dependent polarity of words: An information retrieval approach," *Information Processing & Management*, vol. 53, no. 5, pp. 1062-1079, 2017.
[29] A. Yousefpour, R. Ibrahim, and H. N. A. Hamed, "Ordinal-based and frequency-based integration of feature selection methods for sentiment analysis," *Expert Systems with Applications*, vol. 75, pp. 80-93, 2017.
[30] G. Vinodhini and R. M. Chandrasekaran, "A sampling based sentiment mining approach for e-commerce applications," *Information Processing & Management*, vol. 53, no. 1, pp. 223-236, 2017.
[31] T. Chen, R. Xu, Y. He, and X. Wang, "Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN," *Expert Systems with Applications*, vol. 72, pp. 221-230, 2017.
[32] M. Fernández-Gavilanes, T. Álvarez-López, J. Juncal-Martínez, E. Costa-Montenegro, and F. Javier González-Castaño, "Unsupervised method for sentiment analysis in online texts," *Expert Systems with Applications*, vol. 58, pp. 57-75, 2016.
[33] S. M. Nowlis, B. E. Kahn, and R. Dhar, "Coping with ambivalence: The effect of removing a neutral option on consumer attitude and preference judgments," *The Journal of Consumer Research*, vol. 29, no. 3, pp. 319-334, 2002.
[34] T. Tang, E. Fang, and W. Feng, "Is neutral really neutral? The effects of neutral user-generated content on product sales," *Journal of Marketing*, Article vol. 78, no. 4, pp. 41-58, 2014.
[35] K. Gasper and J. Hackenbracht, "Too busy to feel neutral: Reducing cognitive resources attenuates neutral affective states," *Motivation and Emotion*, vol. 39, no. 3, pp. 458-466, 2014.
[36] M. Koppel and J. Schler, "The importance of neutral examples for learning sentiment," *Computational Intelligence*, vol. 22, no. 2, pp. 100-109, 2006.
[37] S. Baccianella, A. Esuli, and F. Sebastian, "SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining," in *International Conference on Language Resources and Evaluation (LREC)*, 2010, vol. 10, pp. 2200-2204.
[38] E. Cambria, S. Poria, R. Bajpai, and B. Schuller, "SenticNet 4: A semantic resource for Sentiment Analysis based on conceptual primitives," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Osaka, Japan, 2016, pp. 2666-2677.
[39] C. D. Manning, P. Raghavan, and H. Schuetze, "Naïve Bayes text classification," in *Introduction to Information Retrieval*: Cambridge University Press, 2008, pp. 234-265.
[40] M. Bramer, "Nearest neighbour classification," in *Principles of Data Mining*. London: Springer-Verlag, 2007, pp. 31-38.
[41] S. Menard, *Logistic Regression: From Introductory to Advanced Concepts and Applications*. Los Angeles: SAGE, 2010.
[42] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *Journal of Machine Learning Research*, vol. 7, pp. 551-585, 2006.

[43] L. Rokach and O. Maimon, *Data Mining with Decision Trees: Theory and Applications*. World Scientific Publishing Company, 2007.
[44] C. Campbell and Y. Ying, *Learning with Support Vector Machines*. Morgan & Claypool, 2011.
[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1: MIT Press, 1986, pp. 318-362.
[46] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123-140, 1996.
[47] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.
[48] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189-1232, 2001.
[49] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3-42, 2006.
[50] J. Zhu, H. Zou, S. Rosset, and T. Hastie, "Multi-class AdaBoost," *Statistics and Its Interface*, vol. 2, pp. 349-360, 2009.
[51] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.
[52] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," presented at the 3rd International Conference on Learning Representations, San Diego, USA, 2015.
[53] Yelp. (2017, February 5). *Yelp dataset challenge: Round 9 of the Yelp dataset challenge: Our largest yet!* Available: https://www.yelp.com.au/dataset_challenge
[54] J. McAuley. (2014, 01 July). *Amazon product data*. Available: http://jmcauley.ucsd.edu/data/amazon/links.html
[55] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA, 2011, pp. 142-150.
[56] G. S. Budhi, R. Chiong, I. Pranata, and Z. Hu, "Predicting rating polarity through automatic classification of review texts," presented at the IEEE Conference on Big Data and Analytics 2017, Kuching, Malaysia, 16-17 November, 2017.
[57] X. Wang, G. Xu, J. Zhang, X. Sun, L. Wang, and T. Huang, "Syntax-directed hybrid attention network for aspect-level sentiment analysis," *IEEE Access*, vol. 7, pp. 5014-5025, 2019.
[58] M. López, A. Valdivia, E. Martínez-Cámara, M. V. Luzón, and F. Herrera, "E2SAM: Evolutionary ensemble of sentiment analysis methods for domain adaptation," *Information Sciences*, vol. 480, pp. 273-286, 2019.
[59] M. Hur, P. Kang, and S. Cho, "Box-office forecasting based on sentiments of movie reviews and independent subspace method," *Information Sciences*, vol. 372, pp. 608-624, 2016.
[60] L. Zhang, L. Jiang, C. Li, and G. Kong, "Two feature weighting approaches for naïve Bayes text classifiers," *Knowledge-Based Systems*, vol. 100, pp. 137-144, 2016.
[61] L. Gui, Y. Zhou, R. Xu, Y. He, and Q. Lu, "Learning representations from heterogeneous network for sentiment classification of product reviews," *Knowledge-Based Systems*, vol. 124, pp. 34-45, 2017.
[62] J. D. Zhang and C. Y. Chow, "MOCA: Multi-objective, collaborative, and attentive sentiment analysis," *IEEE Access*, vol. 7, pp. 10927-10936, 2019.
[63] I. Pranata and W. Susilo, "Are the most popular users always trustworthy? The case of Yelp," *Electronic Commerce Research and Applications*, vol. 20, pp. 30-41, 2016.
[64] NLTK. (2019, January 25). *Nltk Package*. Available: http://www.nltk.org/api/nltk.html
[65] C. Bhadanea, H. Dalalb, and H. Doshic, "Sentiment analysis: Measuring opinions," *Procedia Computer Science 45*, pp. 808 – 814, 2015.
[66] Scikit-learn. (2019, March 19). *API Reference*. Available: http://scikit-learn.org/stable/modules/classes.html
[67] Keras. (2019, March 8). *Keras: The Python Deep Learning library*. Available: https://keras.io/
[68] Z. Wang, K. Liu, J. Li, Y. Zhu, and Y. Zhang, "Various frameworks and libraries of machine learning and deep learning: A survey," *Archives of Computational Methods in Engineering*, pp. 1-24, 2019.
[69] S. Hameg, M. Lazri, and S. Ameur, "Using naive bayes classifier for classification of convective rainfall intensities based on spectral characteristics retrieved from SEVIRI," *Journal of Earth System Science*, journal article vol. 125, no. 5, pp. 945-955, July 01 2016.
[70] Z. Hui *et al.*, "Development of novel in silico model for developmental toxicity assessment by using naïve Bayes classifier method," *Reproductive Toxicology*, vol. 71, pp. 8-15, 2017.
[71] S. Wang, L. Jiang, and C. Li, "Adapting naïve bayes tree for text classification," *Knowledge & Information Systems*, vol. 44, pp. 77-89, 2015.
[72] Z. Hu, R. Chiong, I. Pranata, W. Susilo, and Y. Bao, "Identifying malicious web domains using machine learning techniques with online credibility and performance data," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 5186-5194.
[73] L. Jiang, C. Li, S. Wang, and L. Zhang, "Deep feature weighting for naïve bayes and its application to text classification," *Engineering Applications of Artificial Intelligence*, vol. 52, pp. 26-39, 2016.
[74] T. F. Chan, G. H. Golub, and R. J. LeVeque, *Updating Formulae and a Pairwise Algorithm for Computing Sample Variances*. New Haven: Stanford University, 1979.
[75] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.
[76] K. Dramé, F. Mougin, and G. Diallo, "Large scale biomedical texts classification: a kNN and an ESA-based approaches," *Journal of Biomedical Semantics*, vol. 7, pp. 40-53, 2016.
[77] L. Y. Hu, M. W. Huang, S. W. Ke, and C. F. Tsai, "The distance function effect on k-nearest neighbor classification for medical datasets," *SpringerPlus*, vol. 5, pp. 1304-1314, 2016.
[78] T. M. Mengesh, H. J. Cho, H. J. Song, K. Sungsoo, and T. S. Chung, "New approach to continuous k-nearest neighbor monitoring in a directed road network," *Adhoc & Sensor Wireless Networks*, vol. 34, no. 1-4, pp. 307-321, 2016.
[79] Z. Pan, Y. Wang, and W. Ku, "A new general nearest neighbor classification based on the mutual neighborhood information," *Knowledge-Based Systems*, vol. 121, pp. 142-152, 2017.
[80] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu, "Diagnosis of multiple cancer types by shrunken centroids of gene expression," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 10, pp. 6567-6572, 2002.
[81] J. A. Nelder and R. W. M. Wedderburn, "Generalized Linear Models," *Journal of the Royal Statistical Society. Series A (General)*, vol. 135, no. 3, pp. 370-384, 1972.
[82] T. J. Hastie and R. J. Tibshirani, *Generalized Additive Models*. Chapman and Hall/CRC, 1990.
[83] G. H. Dunteman and M.-H. R. Ho, "Generalized Linear Models," in *An Introduction to Generalized Linear Models*: SAGE Publications, Inc., 2011, pp. 2-6.
[84] A. J. Dobson and A. G. Barnett, *An Introduction to Generalized Linear Models*, 3rd ed. Boca Raton: CRC Press, 2008.

[85] T. P. Jurka, "Maxent: An R package for low-memory multinomial logistic regression with support for semi-automated text classification.," *R Journal*, vol. 4, no. 1, pp. 56-59, 2012.
[86] D. D. A. Bui, G. D. Fiol, and S. Jonnalagadda, "PDF text classification to leverage information extraction from publication reports," *Journal of Biomedical Informatics*, vol. 61, pp. 141-148, 2016.
[87] J. Lu, P. Zhao, and S. C. H. Hoi, "Online passive-aggressive active learning," *Machine Learning*, vol. 103, no. 2, pp. 141-183, 2016.
[88] L. Ruhwinaningsih and T. Djatna, "A sentiment knowledge discovery model in Twitter's TV content using stochastic gradient descent algorithm," *TELKOMNIKA*, vol. 14, no. 3, pp. 1067-1076, 2016.
[89] F. Guo, L. Zhang, S. Jin, M. Tigabu, Z. Su, and W. Wang, "Modeling anthropogenic fire occurrence in the boreal forest of China using logistic regression and random forests," *Forests*, vol. 7, no. 11, p. 250, 2016.
[90] K. P. Murphy, *Machine Learning*. MIT Press, 2012.
[91] L. Bottou and O. Bousquet, "The tradeoffs of Large Scale Learning," *Advances in Neural Information Processing Systems*, vol. 20, pp. 161-168, 2008.
[92] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, journal article vol. 1, no. 1, pp. 81-106, March 01 1986.
[93] E. B. Hunt, J. Marin, and P. J. Stone, *Experiments in induction*. New York: Academic Press, 1966.
[94] B. Luo, J. Zeng, and J. Duan, "Emotion space model for classifying opinions in stock message board," *Expert Systems with Applications*, vol. 44, pp. 138-146, 2016.
[95] Z. Xu, P. Li, and Y. Wang, "Text classifier based on an improved SVM decision tree," *Physics Procedia*, vol. 33, pp. 1986-1991, 2012.
[96] S. Abhishek, V. Sugumaran, and D. S. Babu, "Misfire detection in an IC engine using vibration signal and decision tree algorithms," *Measurement*, vol. 50, pp. 370-380, 2014.
[97] B. Izydorczyk and B. Wojciechowski, "Differential diagnosis of eating disorders with the use of classification trees (decision algorithm)." *Archives of Psychiatry & Psychotherapy*, vol. 18, no. 4, pp. 53-62, 2016.
[98] D. Yu, Y. Mu, and Y. Jin, "Rating prediction using review texts with underlying sentiments," *Information Processing Letters*, vol. 117, pp. 10-18, 2017.
[99] Y. S. Shah, L. Hernandez-Garcia, H. Jahanian, and S. J. Peltier, "Support vector machine classification of arterial volume-weighted arterial spin tagging images," *Brain and Behavior*, vol. 6, pp. 1-8, 2016.
[100] J. Sun, H. Fujita, P. Chen, and H. Li, "Dynamic financial distress prediction with concept drift based on time weighting combined with Adaboost support vector machine ensemble," *Knowledge-Based Systems*, vol. 120, pp. 4-14, 2017.
[101] K. Chinniyan, S. Gangadharan, and K. Sabanaikam, "Semantic similarity based web document classification using support vector machine," *The International Arab Journal of Information Technology*, vol. 14, no. 3, pp. 285-293, 2017.
[102] S. L. Lo, R. Chiong, and D. Cornforth, "Using support vector machine ensembles for target audience classification on Twitter," *PLoS ONE*, vol. 10, no. 4, p. e0122855, 2015.
[103] S. L. Lo, D. Cornforth, and R. Chiong, "Identifying the high-value social audience from Twitter through text-mining methods," in *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES 2014)*, Singapore, 2014, pp. 325-339.
[104] C. C. Chang and C. J. Lin, "LIBSVM: A library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1-27, 2011.
[105] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," presented at the Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, 2010. Available: http://proceedings.mlr.press
[106] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, / 2014.
[107] R. Adipranata, G. S. Budhi, and B. Setiahadi, "Automatic classification of sunspot groups for space weather analysis," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 8, no. 3, pp. 41-54, 2013.
[108] G. S. Budhi and R. Adipranata, "Handwritten Javanese character recognition using several artificial neural network methods," *Journal of ICT Research and Applications*, vol. 8, no. 3, pp. 195-212, 2015.
[109] G. S. Budhi and R. Adipranata, "Java characters recognition using evolutionary neural network and combination of Chi2 and backpropagation neural network," *International Journal of Applied Engineering Research*, vol. 9, no. 22, pp. 18025-18036, 2014.
[110] L. Sangjae and Y. C. Joon, "Predicting the helpfulness of online reviews using multilayer perceptron neural networks," *Expert Systems with Applications*, vol. 41, no. 6, pp. 3041–3046, 2014.
[111] R. Gaspar, C. Pedro, P. Panagiotopoulos, and B. Seibt, "Beyond positive or negative: Qualitative sentiment analysis of social media reactions to unexpected stressful events," *Computers in Human Behavior*, vol. 56, pp. 179-191, 2016.
[112] W. Yunfeng *et al.*, "Dysphonic voice pattern analysis of patients in Parkinson's disease using minimum interclass probability risk feature selection and bagging ensemble learning methods," *Computational and Mathematical Methods in Medicine*, vol. 2017, pp. 1-11, 2017, Art. no. 4201984.
[113] Q. Wu, Y. Ye, H. Zhang, M. K. Ng, and S. S. Ho, "ForesTexter: An efficient random forest algorithm for imbalanced text categorization," *Knowledge-Based Systems*, vol. 67, pp. 105-116, 2014.
[114] N. Asbai and A. Amrouche, "Boosting scores fusion approach using front-end diversity and Adaboost algorithm, for speaker verification," *Computers & Electrical Engineering*, vol. 62, pp. 648-662, 2017.
[115] W. Lee, C. H. Jun, and J. S. Lee, "Instance categorization by support vector machines to adjust weights in AdaBoost for imbalanced data classification," *Information Sciences*, vol. 381, pp. 92-103, 2017.
[116] O. González-Recio, J. A. Jiménez-Montero, and R. Alenda, "The gradient boosting algorithm and random boosting for genome-assisted evaluation in large data sets," *Journal of Dairy Science*, vol. 96, no. 1, pp. 614-624, 2013.
[117] G. Napolitano, J. C. Sting, M. Schmid, and R. Viviani, "Predicting CYP2D6 phenotype from resting brain perfusion images by gradient boosting," *Psychiatry Research: Neuroimaging*, vol. 259, pp. 16-24, 2017.
[118] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A survey of deep learning and its applications: A new paradigm to machine learning," *Archives of Computational Methods in Engineering*, pp. 1-22, 2019.
[119] L. M. Rojas‐Barahona, "Deep learning for sentiment analysis," *Language and Linguistics Compass*, vol. 10, no. 12, pp. 701-719, 2016.
[120] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84-90, 2017.
[121] C. K. Dewa, A. L. Fadhilah, and A. Afiahayati, "Convolutional neural networks for handwritten Javanese character recognition," *Indonesian Journal of Computing and Cybernetics Systems (IJCCS)*, vol. 12, no. 1, pp. 83-94, 2018.

[122] Y. Yu, H. Lin, J. Meng, and Z. Zhao, "Visual and Textual Sentiment Analysis of a Microblog Using Deep Convolutional Neural Networks," *Algorithms*, vol. 9, no. 2, 2016.
[123] A. Vieira and B. Ribeiro, "Deep Neural Network models," in *Introduction to deep learning business applications for developers: From conversational bots in customer service to medical image processing*: Apress, 2018.
[124] A. Maas. (2011, 01 July). *Large Movie Review Dataset*. Available: http://ai.stanford.edu/~amaas/data/sentiment/
[125] C. Bhadane, H. Dalal, and H. Doshi, "Sentiment analysis: Measuring opinions," *Procedia Computer Science*, vol. 45, pp. 808-814, 2015.
[126] K. Preston. (2013). *Why are grown women typing like thiiiiiiiiiiis?* Available: https://www.mamamia.com.au/why-do-so-many-people-text-likeee-thiiis/
[127] P. Norvig. (2016, June 01). *How to write a spelling corrector*. Available: https://norvig.com/spell-correct.html
[128] S. Lee, J. Ha, M. Zokhirova, H. Moon, and J. Lee, "Background information of deep learning for structural engineering," *Archives of Computational Methods in Engineering*, vol. 25, no. 1, pp. 121-129, 2017.
[129] K. Mader. (2019, February 25). *Simple CNN*. Available: https://www.kaggle.com/kmader/simple-cnn
[130] L. Baraldi. (2019, February 27). *VGG-16 pre-trained model for Keras*. Available: https://gist.github.com/baraldilorenzo/07d7802847aaad0a35d3
[131] T. Mikolov, G. Corrado, I. Sutskever, K. Chen, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111-3119.
[132] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning*, Beijing, China, 2014, vol. 32, no. 2, pp. 1188-1196.