Analyzing BERT's Performance Compared to Traditional Text Classification Models

Bihi Sabiri[1][a], Amal Khtira[1][b], Bouchra El Asri[1][c] and Maryem Rhanoui[2][d]

[1]*IMS Team, ADMIR Laboratory, Rabat IT Center, ENSIAS, Mohammed V University in Rabat, Morocco*

[2]*Meridian Team, LYRICA Laboratory, School of Information Sciences, Rabat, Morocco*

Abstract: Text classification is a task in natural language processing (NLP) in which text data is classified into one or more predefined categories or labels. Various techniques, including machine learning algorithms like SVMs, decision trees, and neural networks, can be used to perform this task. Other approaches involve a new model Bidirectional Encoder Representations from Transformers (BERT) which caused controversy in the machine learning community by presenting state-of-the-art results on various NLP tasks. We conducted an experiment to compare the performance of different natural language processing (NLP) pipelines and analysis models (traditional and new) of classification on two datasets. This study could shed significant light on improving the accuracy during text classification. We found that using lemmatization and knowledge-based n-gram features with LinearSVC classifier and BERT resulted in the high accuracies of 98% and 97% respectively surpassing other classification models used in the same corpus. This means that BERT, TF-IDF vectorization and LinearSVC classification model used Text categorization scores to get the best performance, with an advantage in favor of BERT, allowing the improvement of accuracy by increasing the number of epochs.

1 INTRODUCTION

Natural language processing (NLP) is concerned with how computers can process and understand human language, which is complex and often ambiguous Abbasiantaeb and Momtazi (2022). This includes not only understanding individual words and phrases, but also semantics and syntax, as well as broader meaning and structure in texts. There are many applications of NLP, ranging from machine translation to speech recognition to text-to-speech synthesis to sentiment analysis in texts. Text classification is a common task in NLP and can be useful in many different applications, such as sentiment analysis, topic detection, and language detection. It involves analyzing a piece of text and assigning it to a predefined category or label based on its context. There are several approaches to text classification, including rule-based, machine-based, and hybrid systems. Rule-based systems use a set of predefined rules to classify text, while machine-based systems use machine learning algorithms to learn from pre-labeled data and make classifications. Hybrid systems combine both approaches Li et al. (2019). The optimal model will depend on the specific characteristics of our data and the requirements of our application. Some commonly-used models for NLP classification include :

Naive Bayes Classifiers: This is a probabilistic model that makes predictions based on the probability of certain events occurring. It is often used for text classification tasks and is particularly effective when the features (e.g., words in a document) are independent of one another.

Support Vector Machines (SVMs): This is a linear model that can be used for classification tasks. It works by finding the hyperplane in a high-dimensional space that maximally separates the different classes.

Logistic Regression : This is a linear model that is commonly used for classification tasks. It works well on a variety of NLP tasks and is relatively easy to implement.

Decision Trees and Random Forests: These are models that make decisions based on a series of rules.

[a] https://orcid.org/0000-0003-4317-568X
[b] https://orcid.org/0009-0007-3316-8044
[c] https://orcid.org/0000-0001-8502-4623
[d] https://orcid.org/0000-0002-0147-8466

They are easy to interpret and can handle both numerical and categorical data. Decision trees are often used for classification tasks in NLP.

Neural Network-Based Models: such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. These are machine learning models that are inspired by the structure and function of the human brain. They are particularly effective at handling large and complex datasets and are often used for a wide range of NLP tasks, including classification. It's generally a good idea to try out a few different models and see which one performs the best on your data. You may also want to try ensemble methods, which combine the predictions of multiple models to improve the overall performance.

BERT or Bidirectional Encoder Representations from Transformers: It is a pre-trained neural network model for natural language processing tasks like text classification, question answering, and language interpretation that was created by Google. BERT is able to be fine-tuned on particular tasks using smaller datasets after being trained on a huge corpus of text data. BERT is widely utilized in industry and has demonstrated state-of-the-art performance on a variety of natural language comprehension tasks.

A very broad range of applications are covered by the rather general word "NLP." Here are the most popular applications:

Sentiment Analysis (SA): It is a type of natural language processing (NLP) task that involves classifying texts or speech into positive, negative, or neutral categories Maree et al. (2023). Machine learning methods are commonly used for SA, and these typically involve training a model on a labeled dataset of texts that have been pre-classified as positive, negative, or neutral. This is known as supervised learning. One challenge of using machine learning for SA is that it can be time-consuming to create and label a large dataset, and the model may not generalize well to texts outside of the domain of the training data. In this paper, the authors propose a method for SA that incorporates knowledge from generic knowledge repositories and explores the impact of different types of linguistic features on the performance of the model. They evaluate their approach on three publicly-available datasets and compare it to state-of-the-art techniques.

Automatic Translation Algorithms: Also known as machine translation, have significantly improved the speed and efficiency of translating texts. These algorithms use advanced techniques, such as statistical machine translation and neural machine translation, to analyze and model the structure and meaning of

texts in order to produce accurate translations. While machine translation is not perfect and may not always produce translations that are as fluent and accurate as those produced by human translators, it has greatly enhanced the ability to translate texts quickly and accurately in a variety of languages.

Chatbots: NLP techniques are widely used to develop chatbots, which are computer programs designed to engage in conversations with human users Fischbach et al. (2023). They can be used for a variety of purposes, including customer service, information retrieval, and entertainment.

In this article, we aim to gain a deeper understanding of text classification models and their loss function whose purpose is to adjust the parameters of the model during training, in order to improve their performance on the given task. We will also investigate alternative approaches from the literature and place our findings in context. The article is structured as follows. In section 2, we briefly review related work. Section 3 describes our Study Methods. In Section 4, we discuss our methodology and the proposed pipeline. Section 5 presents the experimental results of the proposed models. Finally, section 6 concludes the paper and suggests potential future research directions for further development of the model.

2 THE STATE OF THE ART

NLP and deep learning methods have been applied in various ways to classify the sentiment of social media posts related to COVID-19 Fernandes et al. (2023). Sentiment analysis is a technique used to determine the emotional tone of a piece of text, and it can be useful in understanding how people are reacting to and discussing a particular event or topic, such as the COVID-19 pandemic. There are many ways to approach sentiment classification, but some common approaches involve using machine learning algorithms to analyze text data and classify it as positive, negative, or neutral in sentiment. Deep learning methods, which are a type of machine learning that involves training artificial neural networks on large amounts of data, have also been used in sentiment classification tasks, as they can often achieve high levels of accuracy.

Bi-LSTM for Sentiment Classification: For example, Arbane et al. (2023) used Social media-based COVID-19 sentiment classification model using Bi-LSTM .

Textual Analytics: Samuel et al. (2020) used an approach to analyze fear and sentiment related to COVID-19 using textual analytics. Its objective was

to collect a large dataset of social media posts, news articles, or other texts that mention COVID-19 and use natural language processing techniques to identify patterns and trends in the language used to describe the pandemic.

Medical Notes Using NLP: A group of researchers Fernandes et al. (2023) was interested in using natural language processing (NLP) to analyze clinical notes in order to identify and classify neurologic outcomes. By developing an NLP algorithm that can accurately process free text clinical notes, they hoped to facilitate the conduct of larger scale studies on neurologic outcomes. NLP is a field of computer science and artificial intelligence that focuses on the interaction between computers and humans through the use of natural language. It involves using computer algorithms to analyze, understand, and generate human language in order to facilitate communication between people and machines. In the context of this research project, the NLP algorithm would be used to analyze clinical notes and identify specific neurologic outcomes mentioned in the notes.

Sentence Level Sentiment Analyzer: Maree et al. (2023) experimentally evaluate four sentiment classifiers, namely support vector machines (SVMs), Naive Bayes (NB), logistic regression (LR) and random forest (RF). Their Findings show that the quality of sentiment analyzers is impacted by the use of various NLP pipelines and semantic linkages. Outcomes specifically show that linking lemmatization and knowledge-based n-gram features produced results with improved accuracy.

3 STUDY METHODS

Text analysis can be defined as the practice of using text processing and NLP techniques to identify the direction of meaning in a text Doaa Mohey (2018). Researchers use various NLP techniques, combining text analysis with the use of outside resources to help identify the semantic orientation of sentences. These methods include machine learning approaches and lexicon-based techniques Pak and Paroubek (2010); Du et al. (2019); Meškelė and Frasincar (2020); Chen et al. (2019); ?. For example, in Doaa Mohey (2016), the polarity of texts at the document level was classified by the authors using artificial neural networks (ANN) and support vector machines (SVM). In particular when using unbalanced data settings, the authors claim that SVM was found to be less effective than ANNs. However, the experiments showed some limitations, especially in terms of the high computational cost required for processing and analyzing

large documents. Therefore, as mentioned in reference Haddi et al. (2015), working at the document level is expensive and inaccurate in some scenarios. This is mainly because the entire document is treated as a single unit and the entire content of the document is classified as BUSINESS, SPORTS, CRIME or SCIENCE with respect to a specific issue. Due to the possibility that two sentences inside a paragraph may have distinct ranking polarities, this problem also occurs while working at the paragraph level. As a result, some researchers Du et al. (2019) have concentrated on text analysis at the sentence level. Researchers that have examined text polarity at the word level can also be found working in the same field Chen et al. (2019). For instance, the authors extracted word polarities by combining the reinforcement learning approach with a long-term memory model (LSTM). In most situations, this method was successful in correctly identifying word polarity, but it occasionally fell short. When examining related works, it becomes clear that feature selection is the primary factor that has a significant impact on the quality of sentiment analysis results, as demonstrated in Sabbah and Hanani (2023), where the authors compared three different feature selection methods: feature frequency, term frequency - inverse of document frequency (TF-IDF), and feature presence. The SVM classifier was used to evaluate system performance. The outcomes showed that, depending on the input attributes, the SVM classifier's performance varied greatly. In a related piece of work, Alam and Yao (2019) established an approach for comparing the effectiveness of several machine learning classifiers, including SVM, Naive Bayes (NB), and maximal entropy (MaxE).

4 METHODOLGY AND PROPOSED PIPELINE

In this section, we went into great length about our research's methodology. All of the standard machine learning, neural network, and natural language processing techniques that we applied in our dataset have been described. There are various steps involved in determining the classification orientation of a particular text. Data preparation, cleaning, and gathering are the first steps in the first phase. This preprocessing improves the quality of the data by removing noise and inconsistencies, which improves the model's effectiveness.

The second stage aims at extracting features from processed texts. The NLP classifier's training and quality assessment are the main goals of the third

phase. We introduce the specifics of each of these phases in the following sections.

4.1 Dataset and Data Acquisition

An open-source text news dataset from Kaggle Huff-Post (2022) was utilized in this study. The public dataset was produced using web scraping techniques from several search engines. As we are constantly exposed to increasing amounts of news, automated data science methods have been used to organize and analyze all of this data. However, using this data to create an effective text news detection architecture has posed a challenge within the data science community. For experimental evaluation purposes, we have retained only the text and category columns. The first task after data acquisition is to clean the raw data as it normally contains special characters including hashtags, consecutive white spaces, URLs and unnecessary symbols.

4.2 Stop Word Deletion

After this step, we go to the second phase of data processing, that is to say the deletion of the most frequent words called stop words Prachi et al. (2022). A stop word is a word that is so common that it is unnecessary to index it or use it in a search. A stop word is an insignificant word appearing in a text. We oppose it with a full word. The meaning of a word is assessed from its distribution (in the statistical sense) in a collection of texts. A word is said to be "empty" if its distribution is uniform over the texts in the collection. In other words, a word that appears with a similar frequency in each of the texts in the collection is not discriminating because it does not make it possible to distinguish the texts from each other. These words should not use any unnecessary processing time or database space. We may easily get rid of them for this reason by keeping a list of words that we regard as stop words. Since our model data is in English, we preprocess it using the English stopwords library. The stopwords preprocessing method must be used to eliminate noise, speed up and improve the model, and conserve memory.

4.3 Tokenization

Then comes the phase of tokenization which consists of converting the text corpus into a vector representation, either by turning each text into a sequence of integer indexes, with each index corresponding to a token in a dictionary, or by converting each text into a vector where the presence or absence of each to-

ken is represented with a binary value based on word count. A variety of feature extraction methods are used by text categorization models. As described in Maree and Eleyat (2020), for words including negatives like "not important" or "not very important," the size of n-gram tokens as decided by the n-gram tokenization process, for instance, can have a considerable impact on how accurate a sentiment analyzer is. The word "important" and the negation word "not" will be separated if unigram characteristics are used to analyse a sentence that contains the phrase "not important." The negation word "not" and the word "important" will also be divided into two distinct tokens. Likewise, using the bigram properties, the negation word "not" and the word "important" will be split into two distinct tokens,"not very" and "very important". As a result, employing n-grams with n bigger than 3 can take a lot of time. Additional research Maree (2021) have demonstrated that bigrams and trigrams together, depending on extrinsic semantic resources, can yield superior results to unigrams alone. According to the research Maree and Eleyat (2020), incorporating higher-order n-grams such as trigrams that are based on external knowledge resources can more accurately capture contextual information in a sentence than unigram or bigram-based tokenizers. Therefore, it is crucial to evaluate the usage of n-gram tokenization in the process of large-scale text analysis through experimentation. We'll go into more depth about each stage of the suggested pipeline for natural language processing in the sections that follow.

4.4 Stemming and Lemmatization

The removal of derivational affixes is the primary goal of stemming, one of the common morphological analysis processes, which aims to provide a common base form for a given word. Reducing inflectional forms and achieving a single base form for words in text sentences are the objectives of this procedure. Both lemmatization and stemming aim to reduce a word's inflectional forms and occasionally derivationally related forms to a single base form. In this situation, there will be fewer dimensions to represent the various words in the text. As a result, it is possible to represent a word as one word in addition to all of its inflectional variants. For example, the word "change" will be used instead of "changes," "changing, "changed" and "changer". This shortening of words aids in accurately estimating word weights and significance within the text. It is a rule-based word reduction stemmer that uses five phases of word reductions in succession Qorib et al. (2023).

4.5 Strategies for Selecting Features

During the construction of a machine learning model for a real-world dataset, we encounter a number of properties, not all of which are really important. When we train a model with extra unnecessary features, we increase the complexity of the model, reducing its ability to generalize. Thus, feature selection is a critical step in the development of a machine learning model. Its goal is to find the perfect mix of features to create a machine learning model Mahmoud et al. (2023).

This article is particularly interested in the fundamental methods of feature extraction used in NLP to analyze textual similarity. The field of computer science and machine learning known as "natural language processing" (NLP) focuses on teaching computers how to process a large volume of human language input (natural). Briefly, NLP refers to the ability of a computer to understand human language. The requirement for feature extraction methods A predefined set of features from the training data is used by machine learning algorithms to provide results for the test data. The fundamental problem with language processing, however, is that machine learning algorithms cannot work directly on plain text. Therefore, in order to transform a text into a matrix (or a vector) of features, some feature extraction techniques are needed. The most well-liked feature extraction techniques include:

1. Bag-of-Words

2. TF-IDF

4.5.1 Bag-of-Words Model

The simplest way to represent a document is to take into account all the words used for each item without differentiation or dependence. Therefore, the analogy is to consider each text as a "bag" of all the words that it contains without regard to context (order, utilisation, etc).

In actuality, this could be, for instance, a measure of how frequently various words are used.

Therefore, a traditional bag-of-words representation would be one in which each article is represented by a vector of the vocabulary size $\overrightarrow{V}$, and our algorithm 1 would be entered into a matrix made up of all of these N articles that make up the corpus.

4.5.2 TF-IDF Model

At this time, it is not necessary to just think of a word's frequency of appearance in a document as its weight, but also to think about that frequency in terms

---

Algorithm 1: Building the Bag-of-Words Model.

1	**for** *Each data in dataset* **do**
2	Tokenize each sentence to words
3	**for** *Each word in words* **do**
4	Lookup the word in our dictionary to see whether it is there.
5	**if** *(Word exists in our dictionary)* **then**
6	word2count[word] = 1
7	**end**
8	**else**
9	*word2count*[*word*]+ = 1
10	**end**
11	**end**
12	**end**

of how common or uncommon the word is across the hole corpus. TF-IDF (Term Frequency-Inverse Document Frequency) is a method used to determine the importance of a word in a document or a collection of documents. We attempt to give a phrase some lexical relevance within a document. In terms of TF-IDF, a relationship is applied between a document and a group of documents that have common key word patterns. It concerns a quantity to quality lexical relationship across a collection of documents. If a document has a specific instance of a given phrase and that term is uncommon in the document's contents, it has a higher likelihood of being relevant as a response to the query for that term.

The value of TF-IFD (see Equation 3) grows proportionally when a word appears in a document more often and declines as a word appears in fewer documents overall. It consists of two sub-parts, which are:

1. Term Frequency (tf) (see Equation 1)

2. Inverse Document Frequency (idf) (see Equation 2)

The term "term frequency" (tf) is a measure of how frequently a word appears in the current document (relative frequency of term t within document D,) . The length of each document varies, therefore it's feasible that a term will appear significantly more frequently in lengthy documents than in shorter ones. To normalize, the term frequency is frequently divided by the length of the document.

$$\mathrm{tf}(t, D) = \frac{f_{t,d}}{\sum_{t' \in D} f_{t',d}} \tag{1}$$

where :
$f_{t,d}$ is Number of times term t appears in document D
and
$\sum_{t' \in D} f_{t',d}$ is total Number of terms in document D

The inverse document frequency (IDF) is a metric used to determine if a term is common or uncommon across all texts in the corpus. It draws attention to terms that are used in a small number of papers overall, or, to put it simply, words with a high idf score that are uncommon. Idf is a log normalized value that is calculated by taking the logarithm of the overall term and dividing the total number of documents D in the corpus by the number of documents containing the term t.

$$idf(t, D) = \log \frac{\|D\|}{1 + \{d \in D : t \in d\}} \tag{2}$$

where $\|D\|$ The number of documents in the corpus and

$\{d \in D : t \in d\}$ is the number of documents in the corpus that include the term t. By adding 1 to the denominator, it guarantees that the denominator will not be zero, and thus we will get the idf score. Thus,

$$tf - idf\,score = tf(t, D) * idf(t, D) \tag{3}$$

In our studies, we will use Bag of words, CNN and TF-IDF to create a vocabulary for a machine learning model by using a dozen different classification algorithms. Further information is introduced in the section on experiments.

5 EXPERIMENTAL RESULTS AND DISCUSSIONS

5.1 The Benchmark

In this study, the proposed technique is assessed using a corpus with 12675 of News Headlines.

The News dataset has been downloaded for the first experiment from HuffPost. It can be used as a benchmark for many computational language problems. To perform text analysis, a popular supervised learning text classification job, it contains 12695 news headlines (see Fig.1). Each record in the dataset consists of the following attributes. News: News article headlines. Category: Category in which the article was published. The dataset includes 4 different news categories in total.

With respect to many well-known machine learning models, such as Multinomial NB, Linear, Logistic Regression,..., we have compared the behavior of a pre-trained default Bag of words and TF-IDF models.

They employ words taken from a bag of words and TF-IDF model to produce the following : Without doing any pre-processing in dataset, the Fig. illustrates
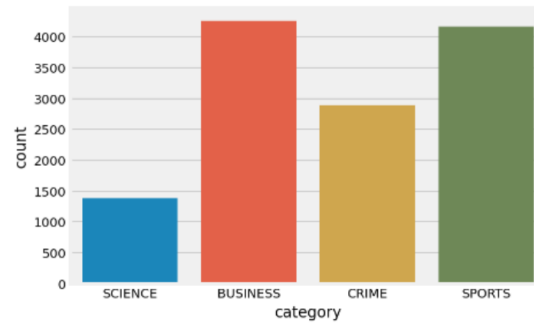

Figure 1: Categorization of News.

the top twenty words used in our dataset, these words, such as "the", "to", "a" "an" and "is" do not add much meaning to the text and can be removed to improve the performance of the model.

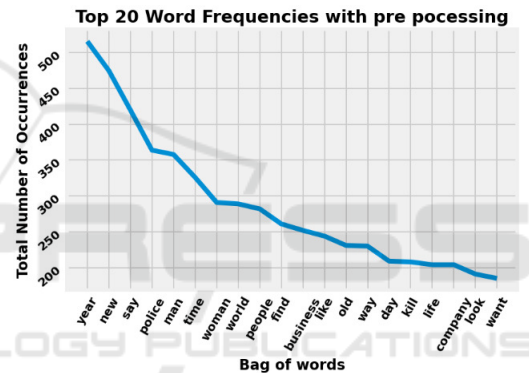Fig.2 shows the Top twenty words after removing these words.

Figure 2: Top 20 words with Preprocessinf.

5.2 Features Extraction with Bag of Words

In the first scenario for our Dataset we produce a bag of uni-gram gram and bi-gram as a process to extracte features that can be used as input to a machine learning model, such as a neural network, to improve its performance.

A multinomial Naive Bayes classifier called MultinomialNB will be the first classifier we train. The second pipeline, which we will build, will make use of the previous pipeline. Then, using LogisticRegression(), we will cross-validate various combinations of hyper-parameters.

Comparing the Uni-gram, Bi-gram classification report bellow (see Table 1) with the raw text classification report, we can see that the model using the preprocessed clean text has some improvement (see Table 2). Therefore, preprocessing (stopword removal, lemmatization) improves the model's performance on this particular task (see Table 2).

Table 1: Perfomances without Preprocessing.

Category	precision	recall	f1-score
BUSINESS	0.69	0.90	0.78
SPORTS	0.95	0.74	0.83
CRIME	0.82	0.88	0.85
SCIENCE	0.92	0.78	0.84
Accuracy	**0.82**		

Table 2: Perfomances with Preprocessing.

Category	precision	recall	f1-score
BUSINESS	0.80	0.88	0.84
SPORTS	0.92	0.83	0.87
CRIME	0.83	0.92	0.87
SCIENCE	0.91	0.81	0.86
Accuracy	**0.86**		

The Accuracy, Precision and F1-score for the four labels are better with clean content (for example, the precision of the category BUSINESS is pushed from 69% to 80% and the accuracy passage is from 82% to 86%).

The performance results for the other models are indicated in the below table. The performance of the other models is evaluated by comparing their accuracies, and the results are shown in the table 3. The model LogisticRegression has the highest accuracy.

Table 3: Accuracies with classification models.

Model	Accuracy
LogisticRegression	0.84
SVC	0.78
KNeighborsClassifier	0.27
DecisionTreeClassifier	0.71
GradientBoostingClassifier	0.76

5.3 Feature Extraction with Sequential Model

In this second stage, we will build a sequential model with multiple layers where each layer of model contains an input and output attribute. We will use feature extraction, which is extremely typical when applying transfer learning in machine learning. Overall, feature extraction is an important step in building a sequential model. It helps to identify important patterns and features in the data, which can improve the performance of the model.

During training, we provide a consensual classi-

fication strategy that we evaluate the performance of these two categories of methods in convolutional neural networks. When necessary, early stopping enables the storage of a model's hyper-parameters. The dropout makes it harder to learn the model as well. In this article, we will construct a sequential model with several layers, each of which has an input and an output attribute. Then, we will use the get layer method to extract the output from the layers. To prevent overfitting of the model, we use two techniques : Early Stopping and Dropout and that are detailed in Sabiri et al. (2022b).

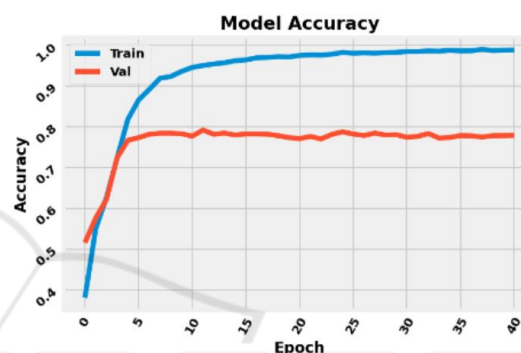After training the model, we obtain this results (see Fig. 3)

Figure 3: Model accuracy curve (training & validation).

Due to the Early Stopping, the model's performance is optimal at a time when accuracy is no longer improving and is beginning to decline. Early stopping requires validation loss to determine if it has decreased. If so, it will create an optimal control point for the current model. In our case, it happens at epoch 35 Sabiri et al. (2022a).

The word "accuracy" is a phrase that lets the training file choose the appropriate metric (binary accuracy Sabiri et al. (2022a), categorical accuracy or sparse categorical accuracy). This choice is based on a number of factors, including the loss functions and the output shape (the shape of the tensor generated by the layer and used as the input by the following layer). For categorization issues, involving more than two classes, categorical_accuracy metric computes the mean accuracy rate across all predictions.

This accuracy allows us to know the percentage of good predictions compared to all predictions. The corresponding Operation is: number of correct answers / total number of guesses (see Table 4).

Table 4: Model Accuracies.

Train	Validation
99%	80%

5.4 Making a TF-IDF Text News Classifier

TF-IDF (Term Frequency - Inverse Document Frequency) (see Section 4.5.2) is a method for extracting features from text data. It is commonly used for natural language processing tasks such as document classification and information retrieval.

The basic idea behind TF-IDF is to weight the importance of each word in a document based on how frequently it appears in the document compared to how frequently it appears in all other documents in the corpus. The weighting is done by computing the product of two terms:

Term Frequency (TF): the number of times a word appears in a document, normalized by the total number of words in the document.

Inverse Document Frequency (IDF): the logarithm of the total number of documents in the corpus divided by the number of documents containing the word.

In this way, the words that are common across all documents are given lower weight, while the words that are unique to a specific document are given higher weight. This makes it possible to identify the most important words in each document, and use them as features for machine learning tasks. By using features extracted by TF-IDF, we train the Dataset with some regression models and the results are chown in Table 5:

If accuracy is high, it means that the model is making correct predictions a large percentage of the time. It is a measure of how well the model is able to correctly identify or classify the inputs it is given. In simple terms, accuracy is the number of correct predictions made by a model divided by the total number of predictions made. It is usually expressed as a percentage. Table 5 shows the models accuracies, which were trained on our dataset.

These results show that the feature extraction with TF-IDf gives better performance than the Bag of words model.

5.5 Text Classification Using LinearSVC, RNN(LSTM), BERT and TF-IDF

The purpose of this second experimental section is to compare the top text classification model from the first section with other text classification models, and explain the underlying principles. The dataset used in this section comes from BBC News (see noa (2019)).

It is a public dataset consisting of 2225 articles.

Table 5: Accuracies with classification models: tf-idf.

Model	Accuracy	Time
LogisticRegression	0.85	3.51s
LinearSVC	0.87	0.31s
LinearSVC L1-based	0.83	0.43s
Multinomial NB	0.86	0.26s
Bernoulli NB	0.84	0.25s
RidgeClassifier	0.86	0.33s
AdaBoost	0.60	1.21s
Perceptron	0.84	0.28s
Passive-Aggresive	0.86	0.32s
NearestCentroid	0.83	0.27s
GradientBoostingClassifier	0.76	19.48s
KNeighborsClassifier	0.68	0.33s
RandomForestClassifier	0.78	6.53s
XGBClassifier	0.73	22.03s
LGBMClassifier	0.74	3.50s
GaussianNB	0.78	8.04s
LSTM	0.90	35.10s

Each falls into one of his five categories (Economy, entertainment, politics, sports, technology).

Two metrics accuracy and training time will be used to assess the algorithms. There will be many different ways covered, each of which is more complex than the last and follows the development of the subject over time. Starting with a linearSVC model methodology (model that is most competitive from the previous scenario), we will use a "conventional" machine learning technique. By employing a Recurrent Neural Network (RNN) technique and encoding words using the Word2Vec algorithm, we will then dig into deep learning. The BERT model, or a condensed variant of it called distil-BERT, will then be used and customized for our particular classification purpose.

Multiclass Classification: SVM does not by default support multiclass classification. It makes binary classification and dividing data points into two classes easier. The same approach is then used to divide the multiclass classification problem into multiple binary classification problems. A single SVM can distinguish between two classes and perform binary classification. In order to categorize data points from the p classes data set using the two breakdown approaches:

- The One-to-Rest approach involves utilizing p SVMs, each of which predicts membership in one

of the p classes (This approach is used in LinearSVC model).

- In contrast, the One-to-One approach utilizes $\frac{p(p-1)}{2} SVMs$.

After the model has been trained, we evaluate its accuracy by comparing its predictions to the labels for both the training data and the test data. We compare it for both, and find that they are almost identical with very high values, the model is not overfitted as a result. (see Table 6).

In summary, linear algorithms converge quickly to reach the "exact" solution, but once the model is optimized, there is not enough chance of improvement.

Table 6: Accuracies for LinearSVC models.

Model	Accuracy train	Accuracy test	Time
LinearSVC	0.99	0.98	2.80s

RNN Deep Learning: Recurrent Neural Networks (RNNs) process words in a specific order in texts while taking into account the word order. It has been found that RNNs have a limit on "how far they can remember" the effect of word correlations without any "internal memory." In machine learning, Long Short Term Memory (LSTM) cells have taken over the role of traditional RNNs to reduce this risk Marinho et al. (2023). LSTM cells provide a kind of memory through internal variables. In most cases, this is appropriate, even if the computational costs are slightly higher. This model utilizes word embeddings through the use of Word2Vec, a technique that was introduced in 2013 and has had a significant impact on natural language processing Belinkov et al. (2020). It involves representing a word as a vector of a specific size, usually 100 or 300, based on its context in text, including the words around it. Word embeddings are usually trained on large text corpora, such as millions of Google News articles for Word2Vec and Wikipedia for a similar method called GloVe. Words with similar meanings are typically located close to each other in this space. The use of RNN in a sequential manner allows for consideration of the order of words, but it also slows down the training process because it cannot be processed in parallel computing. The model is built using the following steps:

1. The input layer is used to specify the format of the input data, so the model knows what to expect.

2. The input, which is a sequence of word indices, is transformed into a sequence of embedded words using the Word2Vec matrix.

3. An LSTM layer that processes the input in both

forward and backward directions is used to maximize the amount of information available. The output from this layer is the last word from the forward LSTM and the backward LSTM, which is a vector of size 30.

4. A dropout layer is used for regularization.

5. A dense layer with 64 neurons and a relu activation function is used to solve the specific problem of classification.

6. A dense layer with a softmax activation function is used to produce a probability distribution for each label.

By comparing the model's predictions to the labels for both the training data and the test data after it has been trained, we can assess the model's accuracy (see Table 7). We compare it for the two and discover that they are nearly equal with high values. As a result, the model is not overfitted.

Table 7: Accuracies LSTM for model.

Model	Accuracy train	Accuracy test	Time
LSTM	0.94	0.93	15.60s

Bert: A pre-trained transformer-based neural network model called BERT (Bidirectional Encoder Representations from Transformers) Devlin et al. (2019) is used for language inference, question answering, and other natural language processing applications. BERT is pre-trained on a huge corpus of text data, allowing it to be fine-tuned for certain applications with less labeled data. Instead of only focusing on the words themselves, BERT is made to understand the context of words in a phrase by taking into account the words before and following them. This bidirectional method aids BERT in achieving cutting-edge performance on a variety of NLP tasks. BERT is based on Google researchers' 2017 publication "Attention Is All You Need," which introduced the transformer design. Instead of processing phrases or paragraphs one word at a time like typical RNN-based models, BERT processes complete sentences or paragraphs at once. This enables BERT to comprehend the context and connections between words in a phrase better.

The two pre-training tasks for BERT are next sentence prediction and masked language modeling. BERT is trained to anticipate the missing words in a phrase given the surrounding context in masked language modeling. BERT is trained to determine whether a specific sentence is the following sentence in a specific pair of sentences.

By learning the connections between words and phrases through these pre-training activities, BERT can excel at downstream tasks like sentiment analysis and question answering.

BERT models are available in a range of sizes, including BERT-base and BERT-large. BERT-large has 340 million parameters, compared to 110 million for BERT-base. Although the larger model offers more fine-tuning options, it also consumes more processing power to run.

BERT is widely used in both business and academics. Hugging Face, Microsoft, and Google are just a few businesses that have made pre-trained BERT models available that may be customized for particular purposes. Other pre-trained models like GPT and XLNet were created as a result of the transformer design and pre-training strategy of BERT.

We see that this complex model performs less accurately than the LinearSVC model for this (simple) problem with limited training data and requires more than 90 times as long to train! (see Table 8)

Table 8: Accuracies for BERT model.

Model	Accuracy train	Accuracy test	Time
BERT	0.96	0.97	241.70s

This may be depressing. My conclusion is that a simple problem does not always require a complex solution, much like with the RNN model.

Our experiment found that the LinearSVC model using TF-IDF vectorization had the highest accuracy, surpassing all other models used in previous research on News headlines analysis.

For the last example with IMBD dataset, which includes 100,000 movie reviews classified as positive or negative, for sentiment analysis. The BERT model yielded the subsequent outcomes 9 :

Table 9: Accuracies for BERT model with another dataset.

Model	Accuracy train	Accuracy test	Time
BERT	0.94	0.93	2H

With the GPU, the training of the model is expected to take about two hours. By running just one epoch, we can already attain a validation accuracy of over 93%. If desired, the accuracy can be further improved by adjusting other parameters and increasing the number of epochs.

6 CONCLUSION

The results of our experiment, which compared a variety of classification models, revealed that LinearSVC, which combined TF-IDF vectorization, had the greatest model accuracy. In this classification problem, using a simple model like LinearSVC provides good accuracy and is more efficient in terms of time compared to more complex models like RNN and BERT. These complex models require significantly more time to run to improve accuracy. However, it can be difficult to improve the performance of a traditional model. On the other hand, complex models like RNN and BERT can be easily improved by experimenting with different architectures or increasing the number of training epochs. It is also noteworthy that the BERT model's accuracy improves with each training epoch, giving the impression that the model is better understanding the text. To improve the evaluation of the BERT model, and validate the accuracies obtained in this study, we plan to improve the text classification by incorporating more data from different disciplines such as : Named entity recognition, Text generation, Language Translation and Sentiment Analysis.

REFERENCES

(2019). BBC News Classification.

Abbasiantaeb, Z. and Momtazi, S. (2022). Entity-aware answer sentence selection for question answering with transformer-based language models. *Journal of Intelligent Information Systems*, 59(3):755 – 777. Cited by: 0.

Alam, S. and Yao, N. (2019). "the impact of preprocessing steps on the accuracy of machine learning algorithms in sentiment analysis, " Computational and Mathematical Organization Theory. 25(3):319–335.

Arbane, M., Benlamri, R., Brik, Y., and Alahmar, A. D. (2023). Social media-based covid-19 sentiment classification model using bi-lstm. *Expert Systems with Applications*, 212. Cited by: 1; All Open Access, Green Open Access.

Belinkov, Y., Bisk, Y., and Pavlick, E. (2020). A comprehensive study of word embedding evaluation methods. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6958–6968.

Chen, R., Zhou, Y., Zhang, L., and Duan, X. (2019). Word-level sentiment analysis with reinforcement learning. *IOP Conference Series: Materials Science and Engineering*, 490(6):062063.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American*

Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1, pages 4171–4186.

Doaa Mohey, E.-D. M. H. (2016). M. *Hussein, Analyzing Scientific Papers Based on Sentiment Analysis, Information System Department Faculty of Computers and Information Cairo University*, 10.

Doaa Mohey, E.-D. M. H. (2018). A survey on sentiment analysis challenges. *Journal of King Saud University - Engineering Sciences*, 30(4):330–338.

Du, C.-H., Tsai, M.-F., and Wang, C.-J. (2019). Beyond word-level to sentence-level sentiment analysis for financial reports. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1562–1566.

Fernandes, M. B., Valizadeh, N., Alabsi, H. S., Quadri, S. A., Tesh, R. A., Bucklin, A. A., Sun, H., Jain, A., Brenner, L. N., Ye, E., Ge, W., Collens, S. I., Lin, S., Das, S., Robbins, G. K., Zafar, S. F., Mukerji, S. S., and Brandon Westover, M. (2023). Classification of neurologic outcomes from medical notes using natural language processing. *Expert Systems with Applications*, 214:119171.

Fischbach, J., Frattini, J., Vogelsang, A., Mendez, D., Unterkalmsteiner, M., Wehrle, A., Henao, P. R., Yousefi, P., Juricic, T., Radduenz, J., and Wiecher, C. (2023). Automatic creation of acceptance tests by extracting conditionals from requirements: Nlp approach and case study. *Journal of Systems and Software*, 197. Cited by: 0.

Haddi, E., "Sentiment Analysis: Text Pre-Processing, R. V., and Domains, C. (2015). *" pp.* 1–133.

HuffPost (2022). News Category Dataset.

Li, W., Gao, S., Zhou, H., Huang, Z., Zhang, K., and Li, W. (2019). The automatic text classification method based on bert and feature union. volume 2019-December, page 774 – 777. Cited by: 25.

Mahmoud, A. M., Desuky, A. S., Eid, H. F., and Ali, H. A. (2023). Node classification with graph neural network based centrality measures and feature selection. *International Journal of Electrical and Computer Engineering*, 13(2):2114 – 2122. Cited by: 0; All Open Access, Gold Open Access.

Maree, M. (2021). "semantics-based key concepts identification for documents indexing and retrieval on the web. *" International Journal of Innovative Computing and Applications*, 12(1):1–12.

Maree, M. and Eleyat, M. (2020). "semantic graph based term expansion for sentence-level sentiment analysis, " international journal of computing. 19(4):647–655.

Maree, M., Eleyat, M., Rabayah, S., and Belkhatir, M. (2023). A hybrid composite features based sentence level sentiment analyzer. *IAES International Journal of Artificial Intelligence (IJ-AI)*, 12(1):284.

Marinho, F. P., Rocha, P. A. C., Neto, A. R. R., and Bezerra, F. D. V. (2023). Short-term solar irradiance forecasting using cnn-1d, lstm, and cnn-lstm deep neural networks: A case study with the folsom (usa) dataset. *Journal of Solar Energy Engineering, Transactions of the ASME*, 145(4). Cited by: 0.

Meškelė, D. and Frasincar, F. (2020). Aldonar: A hybrid solution for sentence-level aspect-based sentiment analysis using a lexicalized domain ontology and a regularized neural attention model. *Information Processing & Management*, 57(3):102211.

Pak, A. and Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Prachi, N. N., Habibullah, M., Rafi, M. E. H., Alam, E., and Khan, R. (2022). Detection of fake news using machine learning and natural language processing algorithms. *Journal of Advances in Information Technology*, 13(6):652 – 661. Cited by: 0; All Open Access, Gold Open Access.

Qorib, M., Oladunni, T., Denis, M., Ososanya, E., and Cotae, P. (2023). Covid-19 vaccine hesitancy: Text mining, sentiment analysis and machine learning on COVID-19 vaccination Twitter dataset. *Expert Systems with Applications*, 212:118715.

Sabbah, A. F. and Hanani, A. A. (2023). Self-admitted technical debt classification using natural language processing word embeddings. *International Journal of Electrical and Computer Engineering*, 13(2):2142 – 2155. Cited by: 0; All Open Access, Gold Open Access.

Sabiri, B., Asri, B. E., and Rhanoui, M. (2022a). Impact of hyperparameters on the generative adversarial networks behavior. volume 1, page 428 – 438. Cited by: 0; All Open Access, Hybrid Gold Open Access.

Sabiri, B., Asri, B. E., and Rhanoui, M. (2022b). Mechanism of overfitting avoidance techniques for training deep neural networks. volume 1, page 418 – 427. Cited by: 0; All Open Access, Hybrid Gold Open Access.

Samuel, J., Ali, G. M. N., Rahman, M. M., Esawi, E., and Samuel, Y. (2020). Covid-19 public sentiment insights and machine learning for tweets classification. *Information (Switzerland)*, 11(6). Cited by: 193; All Open Access, Gold Open Access, Green Open Access.