Exploring Transfer Learning and Deep Learning for Sentiment Analysis: A Cross-Domain Analysis of Model Performance

Marisha Dhakal, Fahad Hossain, Md Ishak, and Marian Temprana Alonso

December 2^{nd}, 2023

1 Introduction:

In contrast to previous generations, we live in a rapidly advancing technological era, where we find ourselves inundated with an unprecedented surge of digital information. These copious amounts of data provide valuable insights into public opinion on a wide range of topics, encompassing more significant matters like politics, public policy, and current events, as well as general perceptions regarding media and consumerism such as new movies, service establishments, and consumer products. These opinion statements and product reviews are easily accessible online, appearing in various forms such as social media posts, blog entries, and product or establishment reviews, which provides service providers, politicians, and anyone seeking insights into public sentiment with a vast pool of information to use. Due to the vastness of this information pool, however, it has become impossible to process it manually, which has led to the design and implementation of various models tasked with performing sentiment analysis on online data sources to decipher whether the nature of a given text is positive, negative, or neutral.

Traditional methods rely on lexicons and non-neural network classifiers, using emotional dictionaries and manual labeling to classify sentiment. These methods are straightforward but limited in their ability to capture complex nuances and context within text [1]. Deep learning methods have gained prominence, employing neural network architectures like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to extract features and patterns from textual data. CNNs can capture spatial information, while RNNs are adept at handling sequential data. Hybrid models combining these approaches aim to strike a balance between speed and context awareness [1]. These deep-learning models can achieve high levels of accuracy with respect to sentiment classification, allowing for the processing and interpretation of large amounts of data. However, much like the human brain, these deep neural network models necessitate copious amounts of pre-processed and pre-labeled data in order to be trained effectively and be able to perform with the required accuracy. This has led to the development and implementation of new techniques that aim to minimize the amount of training required for machine learning models. Transfer learning has emerged as a cutting-edge approach, allowing models to leverage knowledge from one domain to improve performance in another. This technique has proven invaluable in enhancing sentiment analysis accuracy, especially when labeled data is scarce. As sentiment analysis continues to evolve, these approaches will play a crucial role in deciphering and understanding the sentiments expressed in the vast amount of textual data generated daily on the internet [1].

In the following sections, we will provide a concise survey of previous approaches and related works in Section 2, go over the problem statement in Section 3, and data insights in Section 4. Further in the report, we go into the details of the algorithms implemented in Section 5 and move on to the key results in Section 6. We conclude this paper by discussing observations in Section 7, challenges in Section 8 and overall conclusions along with future works in Section 9.

2 Related Works:

In the domain of sentiment analysis, the application of deep learning models has gained a considerable amount of attention in recent years, as evidenced by survey papers like [2], [3] and [4], which provide a detailed overview of a variety of techniques employed in the area of Natural Language Processing (NLP).

In [2], the author provides insights into the performance, advantages, and limitations of different deep learning methods, specifically CNN and Long Short Term Memory (LSTM) networks for sentiment analysis in Twitter data. Subsequently, [3] emphasizes the use of CNNs and RNNs, including, also, the implementation of LSTMs into RNNs. This publication further describes diverse approaches to deep learning implementations, especially based on the purpose of the sentiment analysis model, including two general approaches: i) sentence level, which focuses on the classification of sentences into a sentiment category and ii) aspect level, which has the objective of identifying certain aspects of a text (ex. hateful speech identification). Additionally, the authors of the paper [5] provide an overview of deep learning techniques and perform an in depth analysis of how they are being used in sentiment analysis. The authors also discuss how deep learning techniques have transformed sentiment analysis by enabling computers to recognize emotions in text more precisely, resulting in better predictions and insights.

Furthermore, another group of authors discuss about the importance of sentiment analysis and presented a unique approach combining machine learning, supervised learning, and rule-based categorization strategies. On a variety of datasets, which include movie reviews, product reviews, and MySpace comments, they evaluate the effectiveness of this hybrid methodology. They found that, as indicated by micro- and macro-averaged F1 scores, classification effectiveness has significantly improved. The authors also suggest a semi-automatic method where each classifier works in conjunction with the others to increase efficacy overall [6]. In addition to that, The authors dive deep and performed Twitter sentiment analysis as a way to determine how people feel about certain occasions or goods. The majority of recent research in this area focus on extracting sentiment characteristics by examining lexical and syntactic components that are openly conveyed in tweets, like emoticons, exclamation points, and sentiment words. Important thing to note here is that brand new strategy was presented in which they make use of word embeddings produced via unsupervised learning on big Twitter corpora. These word embeddings detect statistical patterns and latent contextual semantic links between words in tweets producing a comprehensive set of sentiment features for tweets. For sentiment classification, they input this feature set into a deep convolutional neural network [7].

In addition to the copious amounts of research that has been conducted regarding the applications of deep learning models to sentiment analysis, there is a large database of works that consider the application of transfer learning to this same objective. A transfer learning-based approach can improve the accuracy of sentiment analysis on social media content by relaxing the constraints of large amounts of labeled data by using pre-trained deep learning models [8]. In paper [8], the authors show that across all three datasets (Yelp Dataset, Wine Review Dataset, Rotten Tomatoes Dataset), the proposed transfer learning models have consistently outperformed the baseline models in terms of accuracy, Hamming loss, and the macro- and micro-average F1-scores. A pre-trained language model like BERT can be refined with transfer learning in sentiment analysis in order to improve the accuracy of the model [9]. In paper [9], the authors utilize BERT's transfer learning ability to enhance the performance of decision-making in sentiment analysis, specifically in the Bangla NLP domain. Overall, different variations of the BERT machine learning model are widely used in literature, including the aforementioned paper, [9], as well as [10], [11], and [12], where the authors explore diverse approaches to transfer learning within the realm of sentiment analysis. Particularly, the paper titled 'A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media' [10] focuses on the use of transfer learning using the BERT base model as the main model. On top of the BERT model, they use different fine-tuning techniques such as inserting CNN layers, inserting Bi-LSTM layers, and other non-linear layers including BERT-based fine-tuning, and ended up concluding that the BERTbase with the addition of a CNN model was the best fine-tuning approach for their problem. Similarly, a paper titled 'A Fine-Tuned BERT-Based Transfer Learning Approach for Text Classification' [11] compares multiple state-of-the-art language models for fake news detection related to COVID-19 on multiple COVID19 datasets. Additionally, [12] evaluates the performance of several existing machine learning models, including two different variations of the BERT model, with respect to sentiment analysis and emphasizes the importance of considering the data-selection at the training level when choosing which model to use. Ideally, the contextual connotation of words in the training data-set should be similar to that of the words in the test data-set for transfer learning. In accordance with the results achieved in [12], the abundance and variety of texts used to train the BERT models create an ample database for accurately classifying various sets of texts of diverse natures.

3 Problem Formulation

The central research questions trying to be addressed in this study are as follows:

- **Comparison of Deep Learning vs. Traditional Methods:** Our primary goal is to assess whether deep learning models outperform traditional machine learning methods in predicting sentiment across the three diverse datasets.
- **Transfer Learning Impact:** We investigate the impact of transfer learning by fine-tuning transformer-based models, such as BERT, on our domain-specific datasets. Specifically, we examine whether pre-trained models adapted to one domain can yield superior results when applied to different domains.
- **Training from Scratch:** We also explore the differences in evaluation metrics when training the same model architecture, particularly transformer-based models, on our datasets from scratch, without relying on pre-trained weights.

4 Data

4.1 Data Analysis

In this project, we use three distinct datasets in order to perform a comparative analysis between several machine learning algorithms. Specifically, our project road-map involves the extraction and pre-processing of the data from three of the following four sources [13], [14], and [15]. The IMDB and Yelp Reviews data sets were created for the publications [16] and [17] respectively. A brief description of the content of the these chosen datasets is provided in Table 1.

Now let us dive deeper into each datasets:

Data-Set Name	Number of Samples	Usability	Number of Classification Labels
COVID19 Tweets	41157	high	3
IMBD Reviews	50000	high	2
Yelp Reviews	560000	high	2

Table 1: Description of chosen datasets for the proposed experiment.

The Corona Dataset is comprised of 41,157 tweets, each uniquely identified by 'UserName' and 'ScreenName', with sentiment classifications ranging across three categories, where 'Positive' sentiments are predominant at 18046 instances. Despite the comprehensive nature of the dataset, it displays an uneven geographical representation with only 35,531 entries specifying a location, the most common being the United States. The tweets are distributed across 44 dates, with the bulk occurring on '20-03-2020'. For our particular project, we exclude all parameters except the text content of the tweets and the sentiment label. However, it should be noted that the geographical imbalance in the data can lead to some bias. This dataset offers a valuable cross-section of public sentiment during the COVID-19 pandemic, ripe for a sentiment analysis, as well as thematic exploration through keyword analysis to glean deeper insights into the collective mindset during this period.

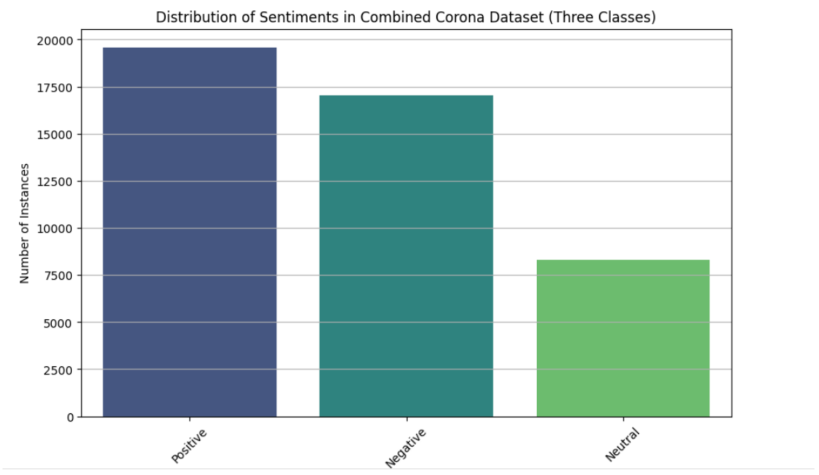


Figure 1: Distribution of text lengths in the corona dataset

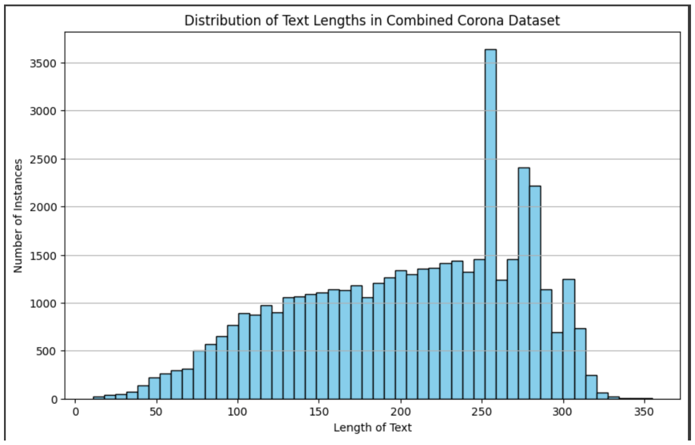


Figure 2: Distribution text lengths of corona dataset

Despite having five main classification labels in the original dataset, we have converted and combined the five main sentiments into three classification levels: Positive, negative and neutral. Figure 1 displays a histogram of sentiments categorized into three classes within the Combined Corona Dataset. It presents 'Positive', 'Negative', and 'Neutral' sentiments on the x-axis, with the number of instances on the y-axis. The 'Positive' category shows the highest number of instances, followed by 'Negative', and then 'Neutral', indicating a larger proportion of positive sentiments in the dataset. This visualization simplifies the sentiment distribution by consolidating it into three main categories, which can be useful for a high-level view of the overall sentiment trends related to COVID-19 communications within the dataset.

Figure 2 illustrates the distribution of text lengths in the dataset. The x-axis represents the length of text, which ranges from 0 to over 350 characters, and the y-axis shows the number of instances. The distribution appears right-skewed, meaning there are more texts with a lower character count and fewer texts as the character count increases. A notable spike occurs at the 250-character mark, suggesting a significant concentration of texts around this length.

For IMDB dataset , it contains 50,000 entries, each with two columns: ‘review’ and ‘sentiment’. The ‘review’ column includes the full text of the movie review, while the ‘sentiment’ column classifies the review as either ‘positive’ or ‘negative’. The dataset is well-suited for sentiment analysis tasks in natural language processing, allowing for the training of models to predict the sentiment of movie reviews or to analyze patterns in public opinion on films. The dataset’s structure is simple and straightforward, which facilitates ease of use in various data analysis applications.

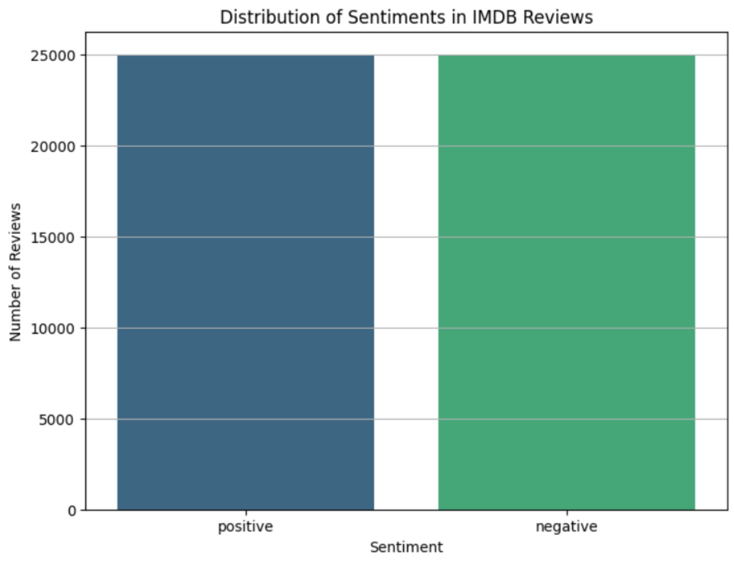


Figure 3: Distribution of sentiments IMDB dataset

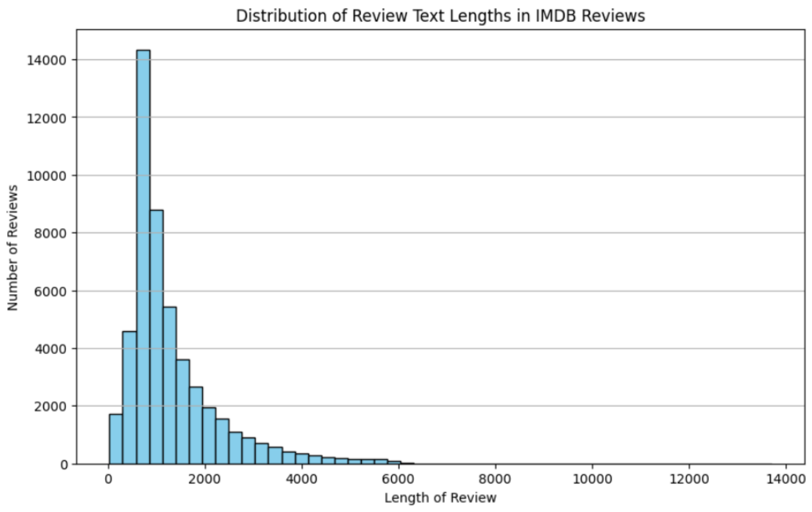


Figure 4: Distribution text lengths of IMDB dataset

Figure 3 demonstrates a bar graph showing the distribution of sentiments in IMDB reviews. There are two bars representing different sentiment categories: positive and negative. The y-axis is labeled ”Number of Reviews,” indicating the count of reviews in each sentiment category, and the x-axis specifies the type of sentiment.

Figure 4 represents the distribution of review text lengths in IMDB reviews. The x-axis shows the length of the reviews, measured in the number of characters, while the y-axis indicates the number of reviews for each length category. The distribution is heavily skewed to the right, with the majority of reviews being of

shorter length and a rapid decline in the number as the review length increases. This suggests that shorter reviews are much more common than longer ones in the IMDB dataset. There are very few reviews with a very high character count, which could be indicative of detailed critiques or in-depth discussions. This type of distribution is common in textual data and can provide insights into user behavior regarding review writing, such as a tendency to write concise reviews.
For Yelp dataset, the training dataset consists of 560,000 entries, each with two columns: ‘text’, which contains the review text, and ‘label’, a binary indicator of sentiment (where ‘0’ represent a negative sentiment and ‘1’ a positive sentiment).

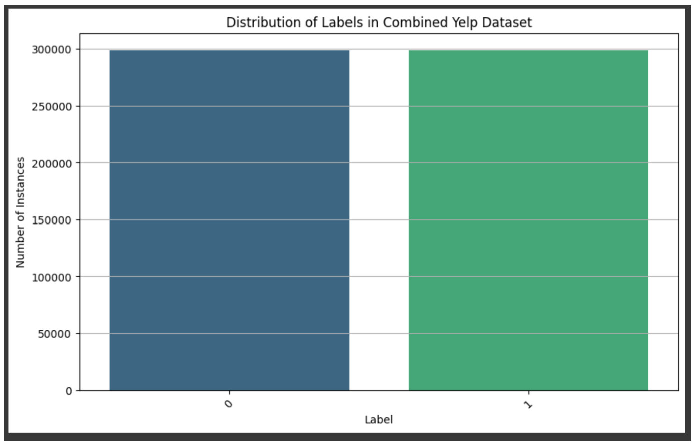


Figure 5: Distribution of labels of YELP dataset

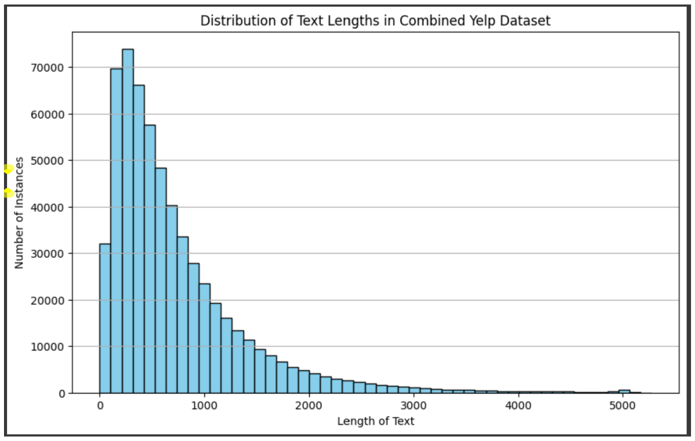


Figure 6: Distribution text lengths of YELP dataset

Figure 5 demonstrates the distribution of labels, representing sentiments or ratings, with two bars representing ‘positive’ and ‘negative’ sentiments. The even distribution suggests that there is a balance between the two categories in this dataset.

Figure 6 illustrates the distribution of text lengths in Yelp reviews. The x-axis indicates the length of the review texts, while the y-axis shows the frequency of reviews for each length. The distribution is right-skewed, with a majority of reviews being shorter, peaking at a lower character count range, and then gradually tapering off as the length increases. This suggests that users tend to leave shorter reviews on Yelp.

4.2 Data Preprocessing

The preprocessing of data holds a crucial role in sentiment analysis as it transforms raw text into a format suitable for analysis and the application of machine learning algorithms. In this section, we describe the various data preprocessing techniques employed in this project. We utilize various libraries such as beautiful soup, regular expression library, lowercase conversion, tokenization, stopwords removal, and lemmatization.

4.2.1 Beautiful Soup:

Reviews or Social media data often contain HTML tags, URLs, emojis, and special characters that can introduce noise and hinder analysis. so, in the initial step of data preprocessing, these elements s are removed from the reviews. This ensures that the text is stripped of any irrelevant or potentially distracting elements. Here is an example:
Original Review: 'The <p>product<p> is amazing! Check it out at www.example.com'
Processed Review: 'This product is amazing! Check it out at'

4.2.2 Regular Expression Library:

The Regular Expression library is utilized to eliminate punctuation, brackets, digits, and unnecessary whitespace, refining the text for subsequent analysis. This step aids in standardizing the text and removing noise. Take this review as an example:
Original Review: Just watched the movie "Inception"! Mind-blowing!!! Highly recommended!!! 10/10
Processed Review: Just watched the movie Inception Mind blowing Highly recommended

4.2.3 Converting to Lowercase:

To ensure uniformity in the dataset, all text is converted to lowercase. Converting all text to lowercase ensures that the model doesn't treat words with different cases as distinct. For instance:
Original Review: "The Performance of the Product was Excellent!"
Processed Review: "the performance of the product was excellent"

4.2.4 Tokenization:

Tokenization involves breaking down sentences into individual words. This step is essential for subsequent analysis where individual words are considered. Consider the following example:
Original Review: "The product is good, but it was too expensive."
Processed Review: ["the", "product", "is", "good", "but", "it", "was", "too", "expensive"]

4.2.5 Removing Stopwords:

Stopwords, common words that often don't carry significant meaning, are removed to focus on essential content. Examples include articles, prepositions, and auxiliary verbs. Consider this review:
Original Review: The movie was good, but it could have been better with more character development.
Processed Review: movie good better character development

4.2.6 Lemmatization:

Lemmatization involves reducing words to their base form. This ensures that variations of a word are treated as a single entity. Consider the following example:

Original Review: The product is running smoothly
Processed Review: the product run smooth

4.3 Feature Extraction

In natural language processing (NLP) tasks like sentiment analysis, effective feature extraction is crucial for transforming raw text data into a format suitable for analysis and model training. In this section, we discuss three prominent techniques employed for feature extraction: Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TFIDF), and Word Embeddings. We applied Bow (Count Vectorizer) and TFIDF models for the Traditional Model and Word Embeddings for the deep-learning model.

4.3.1 Bag of Words:

Bag of Words (BoW) serves as a foundational technique in text analysis by portraying a document as an unstructured collection of words. It disregards grammatical structure and word order while prioritizing the analysis of word frequency. The core goals of employing BoW include:
Frequency Analysis: BoW allows us to analyze the frequency of words in a given text corpus. For example, consider the following tweet: ‘Great movie! The plot was captivating, and the acting was superb.’ Applying BoW, we obtain a representation that emphasizes the frequencies of words like ‘great’, ‘movie’, ‘plot’, ‘captivating’, ‘acting’, and ‘superb’.
Data Transformation: BoW is employed to fit and transform text data into a numerical format suitable for traditional machine learning models. Each document is represented as a vector, where each element corresponds to the frequency of a specific word.
Document-Term Matrix Inspection: The resulting Document-Term Matrix (DTM) provides insights into the occurrence of words across documents. Rows represent documents, columns represent words, and the matrix entries denote word frequencies.

4.3.2 Term Frequency-Inverse Document Frequency (TFIDF):

TFIDF is a refinement of BoW model that considers the importance of words in a document relative to their frequency across a collection of documents. The key purposes of TFIDF include:
Measurement of the importance of a word: TFIDF measures the importance of a word in a specific document. For instance, in a movie review dataset, the word ‘plot’ may have higher TFIDF weight if it is crucial to describing the content.
Normalization of Document Length: TFIDF addresses imbalances in document length by normalizing word frequencies. It also ensures that longer documents do not dominate others solely because of their higher raw frequency.
Suitability for Information Retrieval: TFIDF is well-suited for information retrieval tasks where documents need to be ranked based on relevance to a query.

4.3.3 Word Embeddings:

The Word Embeddings technique serves as a powerful tool for capturing semantic meaning in text data by making it compatible with deep learning models. The process involves several key steps: applying tokenization, padding, and embedding.
Tokenization: The first step in preparing text data for deep learning models is tokenization. This involves breaking down the raw text into smaller units such as words or subwords. Each unit or token becomes a discrete entity which forms the basis for subsequent analysis.
Padding: To ensure uniformity in the length of input sequences, padding is applied. This involves adding

zeros or a specific token to shorter sequences which brings them to a consistent length. This step is crucial for creating batches of input data that can be efficiently processed by deep learning models.
Embeddings: The core of embeddings lies in transforming words into dense vectors that capture semantic relationships. This process enhances the model's ability to understand and generalize from the underlying meaning of words in the data.

5 Experiment

In accordance with the objective of this work, we have implemented a variety of traditional machine learning and deep learning algorithms on the three data-sets described in Section 4. We perform a comparative analysis of these algorithms by evaluating their performance in terms of accuracy for each of the specified data-sets. In this section, we elaborate upon the architectures and hyperparameters of the implemented models. Note that hyperparameter tuning was performed for the traditional algorithms, but not for the deep learning algorithms, due to computational limitations.

5.1 Traditional Algorithms

The traditional machine learning algorithms implemented in this work are: Support-Vector Machine (SVM), Naive Bayes (NB) Classifier, Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), and K-Nearest Neighbors (KNN). Hyperparameter tuning was conducted for each algorithm using scikit-learn's GridSearchCV function. This function systematically explores a predefined set of hyperparameters for each algorithm, evaluating their accuracy through an iterative process. The best hyperparameters were determined by selecting the combinations that yielded the highest accuracy during training. Tuning was carried out individually for each vectorized dataset, considering the TF-IDF and count vectorized versions separately. The accuracy was evaluated using a 5-fold cross-validation scheme. The resulting optimal hyperparameters are listed in Table 2. The best accuracy scores that result from each of these algorithms (after hyperparameter tuning) are provided in Section 6.

Hyperparameters for Traditional ML Models

Model	Data-Set	Vectorizer	Optimal Hyperparameters		
SVM	Corona	TF-IDF	$C = 10$	kernel = 'linear'	
		Count	$C = 1$	kernel = 'linear'	
	IMDB	TF-IDF	$C = 10$	kernel = 'linear'	
		Count	$C = 0.01$	kernel = 'linear'	
	Yelp	TF-IDF	$C = 10$	kernel = 'linear'	
		Count	$C = 0.01$	kernel = 'linear'	
NB	Corona	TF-IDF	alpha = 0.1		
		Count	alpha = 0.5		
	IMDB	TF-IDF	alpha = 0.2		
		Count	alpha = 1.0		
	Yelp	TF-IDF	alpha = 0.2		
		Count	alpha = 1.0		
LR	Corona	TF-IDF	$C = 10$	penalty = 'l1'	solver = 'liblinear'
		Count	$C = 10$	penalty = 'l1'	solver = 'liblinear'
	IMDB	TF-IDF	$C = 100$	penalty = 'l2'	solver = 'lbfgs'
		Count	$C = 1$	penalty = 'l2'	solver = 'sag'

	Yelp	TF-IDF	$C = 100$	penalty = ’l1’	solver = ’saga’
		Count	$C = 1$	penalty = ’l2’	solver = ’lbfgs’
DT	Corona	TF-IDF	max depth = 20 min samples split = 2	min samples leaf = 1	criterion = ’gini’
		Count	max depth = 20 min samples split = 5	min samples leaf = 1	criterion = ’entropy’
	IMDB	TF-IDF	max depth = 20 min samples split = 2	min samples leaf = 2	criterion = ’gini’
		Count	max depth = 20 min samples split = 2	min samples leaf = 1	criterion = ’gini’
	Yelp	TF-IDF	max depth = None min samples split = 2	min samples leaf = 1	criterion = ’gini’
		Count	max depth = None min samples split = 10	min samples leaf = 1	criterion = ’gini’
RF	Corona	TF-IDF	max depth = None min samples split = 5	min samples leaf = 1 num estimators = 50	max features = ’sqrt’
		Count	max depth = None min samples split = 2	min samples leaf = 1 num estimators = 50	max features = ’sqrt’
	IMDB	TF-IDF	max depth = None min samples split = 5	min samples leaf = 1 num estimators = 200	max features = ’sqrt’
		Count	max depth = 20 min samples split = 5	min samples leaf = 1 num estimators = 200	max features = ’sqrt’
	Yelp	TF-IDF	max depth = 20 min samples split = 5	min samples leaf = 1 num estimators = 200	max features = ’sqrt’
		Count	max depth = 20 min samples split = 5	min samples leaf = 1 num estimators = 200	max features = ’sqrt’
KNN	Corona	TF-IDF	num neighboors = 3	p = 2	weights = ’distance’
		Count	num neighboors = 3	p = 2	weights = ’distance’
	IMDB	TF-IDF	num neighboors = 10	p = 2	weights = ’distance’
		Count	num neighboors = 10	p = 2	weights = ’distance’
	Yelp	TF-IDF	num neighboors = 15	p = 2	weights = ’uniform’
		Count	num neighboors = 5	p = 2	weights = ’distance’

Table 2: Optimal hyperparameters resulting from scikit-learn’s GridSearchCV function. Hyperparameters not listed here were not tuned over due to computational limitations. They were set to be the scikit-learn models default values.

5.2 Deep Learning Algorithms

In this subsection, we describe the architecture of the deep learning algorithms that were implemented in this work. Due to computational limitations, hyperparameter tuning was not performed in this algorithms. The same default hyperarameters were employed across all datasets.

5.2.1 Multi-Layer Perceptron

The first deep learning algorithm implemented is a Multi-Layer-Perceptron (MLP). As shown in Figure 8, the structure consists of an input and output layer with three hidden layers nested in between. The number of neurons in the input layer is defined by the structure of the processed data-set. The output layer contains two or three neurons, depending on the number of classes in the dataset (i.e. two neurons for the IMDB and Yelp datasets and three neurons for the Corona dataset). The three hidden layers have 16, 32, and 32 neurons respectively. The activation function implemented was ReLu, with Softmax in the final layer. The optimizer was Adam with a learning rate of $1e-5$. These same parameters were applied for all three datasets. The resulting accuracies are provided in Section 6

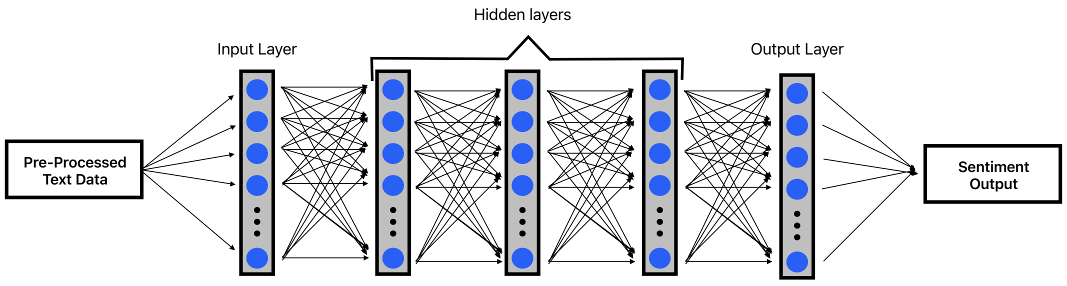


Figure 7: MLP model with three hidden layers

5.2.2 Convolutional Neural Networks

CNN in sentiment analysis leverage specialized layers to automatically learn hierarchical features from text data and enables effective classification of sentiment in natural language. The CNN model for our research is implemented using the Keras library. The model begins with an embedding layer, which is responsible for learning dense vector representations of words in the input data. Following the embedding layer, a one-dimensional convolutional layer (Conv1D) with 64 filters and a kernel size of 3 is applied. This convolutional layer is designed to capture patterns and relationships within the input sequence and then utilizes the rectified linear unit (ReLU) activation function. The subsequent layer is a global max-pooling layer (GlobalMaxPooling1D) which extracts the maximum value from each feature map generated by the convolutional layer. The model then employs two densely connected layers (Dense) with 64 and 32 units, respectively with ReLU activation function. Finally, the output layer consists of two units (IMDB and Yelp dataset) and three units (Corona Dataset) which corresponds to the number of classes for sentiment analysis. The softmax activation function is utilized to transform the network's output into probability distributions. Then it predicts the classification of input sequences into different sentiment categories. The model is trained using the Adam optimizer and binary cross-entropy(IMDB and Yelp dataset) and categorical cross-entropy (Corona Dataset) loss function. Overall, this CNN architecture is tailored for extracting features from input text data for effective sentiment classification.

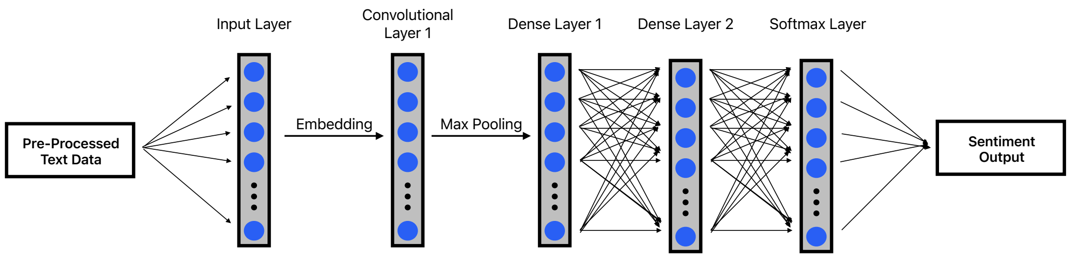


Figure 8: CNN model with three hidden layers

5.3 BERT

BERT, an acronym for Bidirectional Encoder Representations from Transformers, is crafted to pretrain extensive bidirectional representations from unlabeled text, incorporating both left and right context at all layers [18]. BERT was trained on a total of 3.3 billion words, sourced from 2.5 billion from Wikipedia and 0.8 billion from BooksCorpus. Multiple versions of BERT exist, and we have chosen to implement BERT Base Uncased as our third deep learning algorithm. The BERT Base architecture consists of 12 layers (transformer blocks), with 768 hidden units, 12 attention heads, and a total of 110 million parameters.

Our utilization of BERT involves two distinct scenarios. In the first scenario, we engage in transfer learning with BERT by fine-tuning the model on our dataset, specifically for sentiment analysis. In the second scenario, we undertake the ambitious task of training BERT from scratch using our dataset.

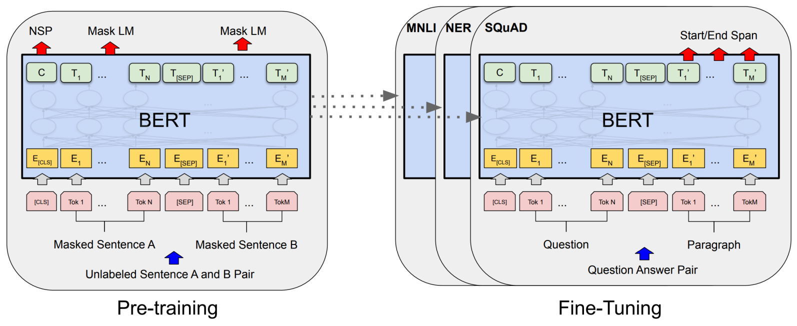


Figure 9: Overall pre-training and fine-tuning procedures for BERT [18]

Notably, aside from the output layers, identical architectures are employed in both pre-training and fine-tuning phases which can be observed in Figure 9. The same set of pre-trained model parameters initializes models for diverse downstream tasks. Throughout the fine-tuning process, all parameters undergo adjustment [18].

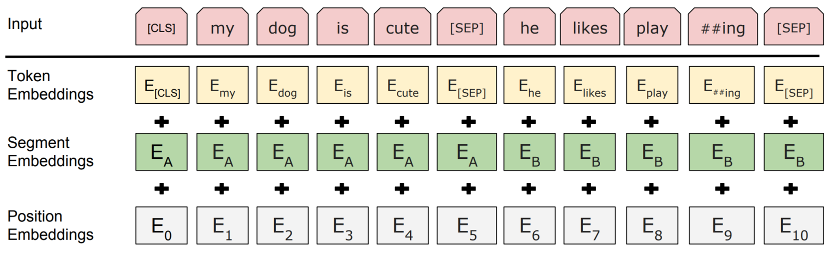


Figure 10: BERT input representation [18]

In the textual input, [CLS] is added at the start and [SEP] at the end of each sentence; [CLS] marks the input example's beginning, while [SEP] serves as a separator distinguishing segments. Input for BERT models are input embedings. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings as observed in figure 10. These components collectively contribute to forming a comprehensive input representation, encompassing token semantics, segment distinctions, and positional information for effective processing by the transformer model [18].

Since BERT is not designed exclusively for classification tasks, a classifier is added on top of the BERT model to enable classification purposes as shown in Figure 11. This additional classifier serves to adapt BERT's contextualized representations to specific classification tasks. By adding this task-specific layer, BERT can be effectively fine-tuned for tasks such as sentiment analysis, question answering, or any other classification task without modifying the underlying BERT architecture.

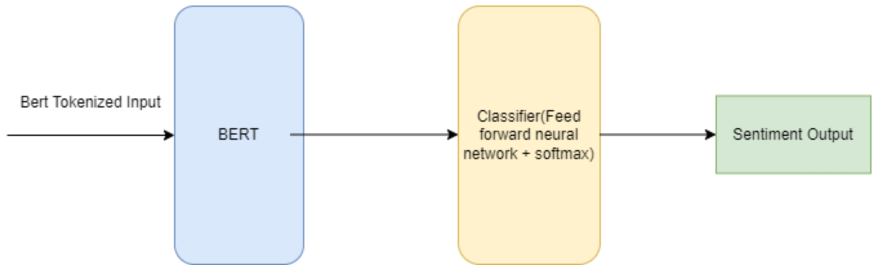
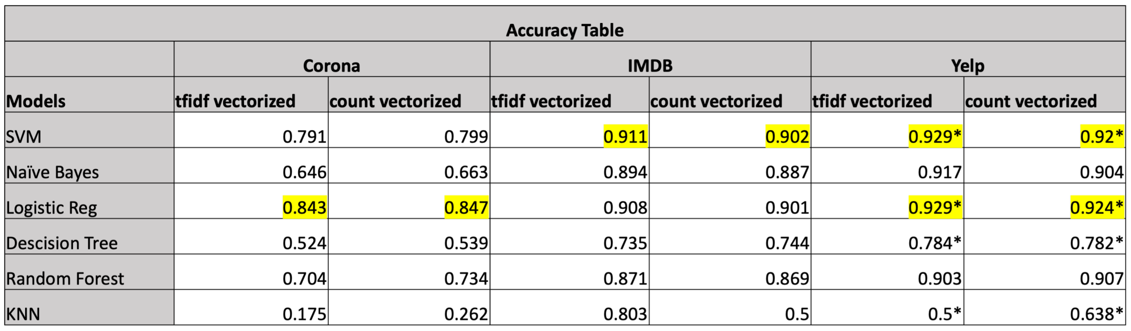


Figure 11: BERT Architecture for Text classification

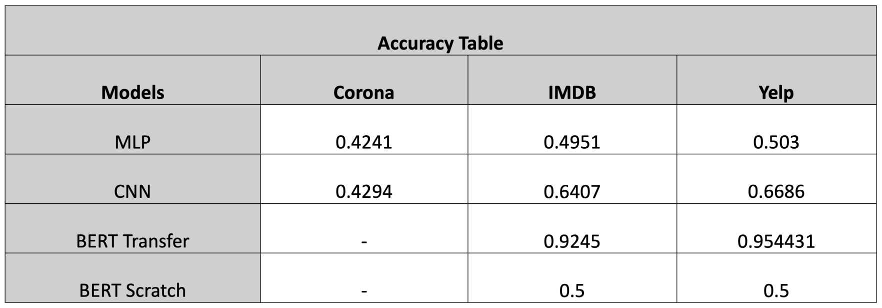
6 Key Results



Accuracy Table						
	Corona		IMDB		Yelp	
Models	tfidf vectorized	count vectorized	tfidf vectorized	count vectorized	tfidf vectorized	count vectorized
SVM	0.791	0.799	0.911	0.902	0.929*	0.92*
Naïve Bayes	0.646	0.663	0.894	0.887	0.917	0.904
Logistic Reg	0.843	0.847	0.908	0.901	0.929*	0.924*
Descision Tree	0.524	0.539	0.735	0.744	0.784*	0.782*
Random Forest	0.704	0.734	0.871	0.869	0.903	0.907
KNN	0.175	0.262	0.803	0.5	0.5*	0.638*

Figure 12: Testing accuracy for the Traditional algorithms.

Note: The accuracy measures followed by an asterisk result from a reduced dataset (due to limitations in computational resources).



Accuracy Table			
Models	Corona	IMDB	Yelp
MLP	0.4241	0.4951	0.503
CNN	0.4294	0.6407	0.6686
BERT Transfer	-	0.9245	0.954431
BERT Scratch	-	0.5	0.5

Figure 13: Testing accuracy for the deep learning algorithms.

The results of our comparative analysis are shown above, where Figure 12 displays the accuracy measures resulting from the traditional algorithms and Figure 13 shows the results for the deep learning algorithms. Due to time and computational limitations, we were not able to obtain the accuracies for the Corona dataset using BERT. Following the trends demonstrated by the other results and the nature of the Corona dataset, we predict that the accuracy for transfer learning would be lower than that for the IMDB and Yelp datasets. Likewise, since the accuracies resulting from training BERT from scratch are equivalent to accuracy from taking one class as default and predicting everything as that class, we predict that the Bert from Scratch accuracy for Corona would be around 0.44, as the class containing the most instances in this dataset (the positive class) comprises about 44% of the total dataset.

Furthermore, we've included Epoch vs. Accuracy plots for both the YELP and IMDB datasets, illustrating the performance of the BERT model in scenarios of training from scratch and transfer learning.

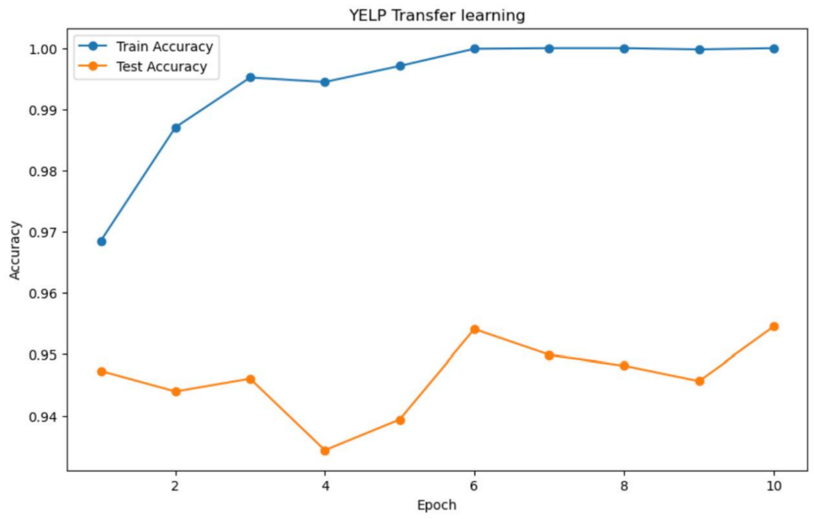


Figure 14: Epoch vs Accuracy for YELP dataset on BERT Transfer Learning.

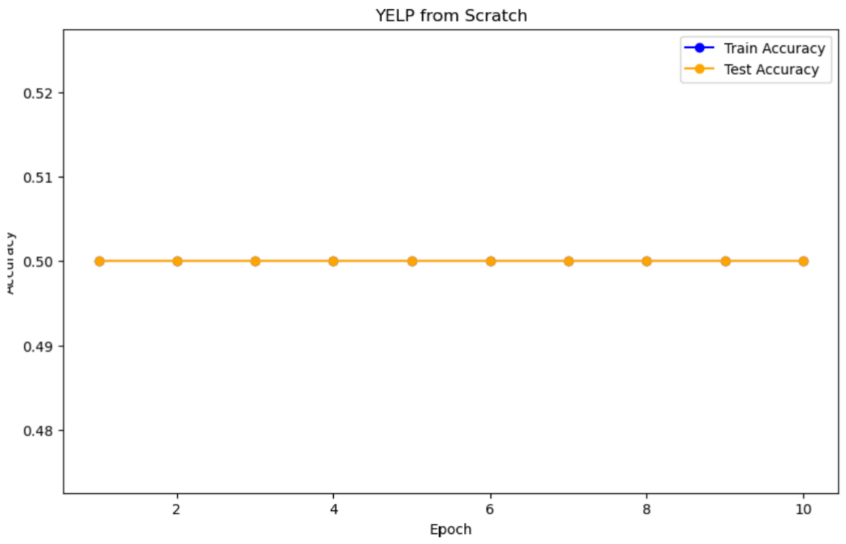


Figure 15: Epoch vs Accuracy for YELP dataset on BERT from scratch.

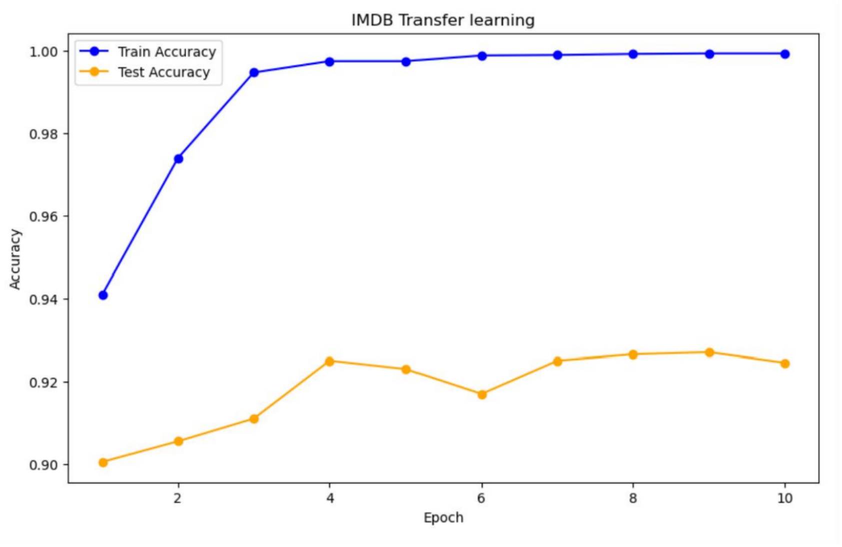


Figure 16: Epoch vs Accuracy for IMDB dataset on BERT Transfer Learning.

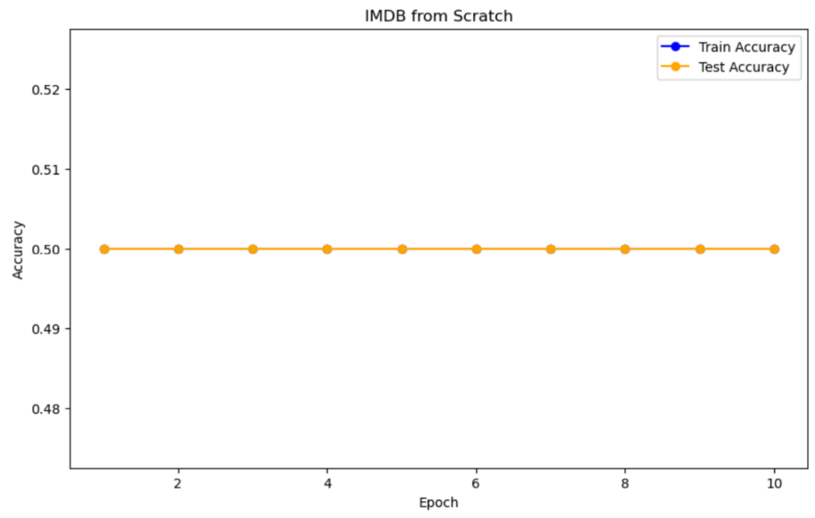


Figure 17: Epoch vs Accuracy for IMDB dataset on BERT from scratch.

The graphs highlight that a small number of epochs is sufficient for effective fine-tuning, and it's evident that a small dataset lacks adequacy for training these models entirely from scratch.

7 Observations

In this section, we will explore the observations derived from the experiment and analyze the results presented in the previous section.

7.1 Traditional Models

Our exploration revealed that Support Vector Machines (SVM) demonstrated commendable performance on both the IMDB and Yelp datasets, showcasing the robustness of traditional models. Logistic Regression emerged as the optimal choice for the Corona dataset. However, it is crucial to note that the presence of dataset imbalance within the Corona dataset significantly impacted the accuracy of traditional models, highlighting the challenge posed by skewed class distributions.

7.2 Deep Learning Models

Contrary to expectations, training deep learning models from scratch did not yield discernible improvements compared to traditional models. It is essential to consider that the dataset primarily comprises textual data, and the underwhelming performance of deep learning models may be attributed to the absence of contextual tokenization. This oversight highlights the critical role of preprocessing techniques, such as contextual tokenization, in enhancing the model's ability to capture nuanced relationships within text data.

7.3 Learning from Scratch

A significant cautionary note surfaced regarding the inadvisability of training extensive models, such as BERT, from the ground up with a limited dataset. The results unequivocally discourage such an approach, emphasizing the necessity of substantial data for effectively training large-scale models.

7.4 Transfer Learning:

A noteworthy observation is the remarkable efficacy of transfer learning, particularly exemplified by BERT. Transfer learning demonstrated superior performance even with limited data, showcasing the efficiency of leveraging pre-existing knowledge encoded in transferable representations. Particularly, transfer learning obviates the need for a high number of epochs, expediting model convergence.

However, it is imperative to acknowledge that the heightened performance achieved by transfer learning models is accompanied by an augmented demand for time and computational resources. The intricacies involved in fine-tuning transfer learning models impose a greater burden on computational infrastructure compared to conventional machine learning models.

8 Challenges

In this section, we will discuss the challenges encountered during the implementation of the algorithm in our selected datasets.

- **Limited Dataset Size:** Dealing with a limited dataset for deep learning tasks posed a challenge, making effective model training difficult and leading to suboptimal performance.
- **Imbalanced Dataset:** Handling class imbalance in the Corona dataset presented a challenge, impacting the model's ability to generalize well, especially on minority classes.
- **Textual Data Complexity:** Navigating the complexity of textual data in deep learning models required advanced preprocessing, such as contextual tokenization, to capture nuanced relationships, posing a significant challenge.

- **Resource Intensiveness of Deep Learning Models:** Managing the computational intensity of training models like BERT strained our limited computational resources, affecting the feasibility of the experiment to overcome which, the size of the dataset had to be decreased.
- **Time and Resource Demands of Transfer Learning:** Handling the significant time and computational resource demands of transfer learning models presented practical challenges in terms of experiment logistics.
- **Model Selection and Hyperparameter Tuning:** Navigating the intricacies of model selection and hyperparameter tuning presented challenges, as inadequate choices resulted in diminished model performance in regards to the traditional algorithm.
- **Balancing Model Performance and Resource Constraints:** Striking a balance between achieving superior model performance and managing computational constraints was a crucial challenge, requiring careful consideration and optimization.

9 Conclusions and Future Directions

In this project, we perform a comparative analysis between traditional machine learning, deep learning, and transfer learning with respect to sentiment analysis. In this work, we find that, within the category of traditional machine learning algorithms, SVM and Logistic Regression give highly accurate results across all datasets in comparison with the other traditional machine learning models despite class imbalances. Additionally, it can be noted that the deep learning algorithms do not necessarily outperform the traditional algorithms, which raises the question of whether it is always suitable to implement deep learning methods for all kinds of tasks. Particularly, in this project, the traditional machine learning algorithms perform better with regard to accuracy than the deep learning models. It should be noted that the preprocessing methods implemented in the case of the MLP and CNN models does not allow for considering the context of the words, which might be a reason for the relatively low performance of these models. Another notable conclusion is that if a problem can be solved utilizing a large deep learning model available for fine-tuning (transfer learning), the resulting accuracy outperforms that of the traditional machine learning algorithms as well as the deep learning models trained from scratch.

Overall, this lays the foundation for future works where we focus on improving the data fed into the models by considering methods that preserve context, expanding the datasets, optimizing the model training (hyperparameter tuning), and further exploring transfer learning, along with other model architectures. Additionally, efficient resource management remains a key area for future advancement in sentiment analysis.

References

[1] R. Liu, Y. Shi, C. Ji, and M. Jia, "A survey of sentiment analysis based on transfer learning," *IEEE Access*, vol. 7, pp. 85 401–85 412, 2019.

[2] D. Goularas and S. Kamis, "Evaluation of deep learning techniques in sentiment analysis from twitter data," in *2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML)*. IEEE, 2019, pp. 12–17.

[3] M. I. Prabha and G. Umarani Srikanth, "Survey of sentiment analysis using deep learning techniques," in *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*, 2019, pp. 1–9.

[4] J. Yuan, Y. Wu, X. Lu, Y. Zhao, B. Qin, and T. Liu, "Recent advances in deep learning based sentiment analysis," *Science China Technological Sciences*, pp. 1947–1970, 2020. [Online]. Available: https://doi.org/10.1007/s11431-020-1634-3

[5] L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1253, 2018.

[6] R. Prabowo and M. Thelwall, "Sentiment analysis: A combined approach," *Journal of Informetrics*, vol. 3, no. 2, pp. 143–157, 2009.

[7] Z. Jianqiang, G. Xiaolin, and Z. Xuejun, "Deep convolution neural networks for twitter sentiment analysis," *IEEE access*, vol. 6, pp. 23 253–23 260, 2018.

[8] J. Tao and X. Fang, "Toward multi-label sentiment analysis: a transfer learning based approach," *Journal of Big Data*, vol. 7, pp. 1–26, 2020.

[9] N. J. Prottasha, A. A. Sami, M. Kowsher, S. A. Murad, A. K. Bairagi, M. Masud, and M. Baz, "Transfer learning for sentiment analysis using bert based supervised fine-tuning," *Sensors*, vol. 22, no. 11, p. 4157, 2022.

[10] M. Mozafari, R. Farahbakhsh, and N. Crespi, "A bert-based transfer learning approach for hate speech detection in online social media," in *Complex Networks and Their Applications VIII*, H. Cherifi, S. Gaito, J. F. Mendes, E. Moro, and L. M. Rocha, Eds. Cham: Springer International Publishing, 2020, pp. 928–940.

[11] R. Qasim, W. H. Bangyal, M. A. Alqarni, A. Ali Almazroi *et al.*, "A fine-tuned bert-based transfer learning approach for text classification," *Journal of healthcare engineering*, vol. 2022, 2022.

[12] A. d. Arriba, M. Oriol, and X. Franch, "Applying transfer learning to sentiment analysis in social media," in *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, 2021, pp. 342–348.

[13] A. Miglani, "Coronavirus tweets nlp - text classification," 2020. [Online]. Available: https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification/data

[14] L. N, "Imdb dataset of 50k movie reviews," 2011. [Online]. Available: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

[15] H. Face, "Yelp reviews sentiment dataset," 2015. [Online]. Available: https://www.kaggle.com/datasets/thedevastator/yelp-reviews-sentiment-dataset

[16] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," Portland, Oregon, USA, pp. 142–150, June 2011. [Online]. Available: http://www.aclweb.org/anthology/P11-1015

[17] X. Zhang, J. Zhao, and Y. LeCun, "Character-level Convolutional Networks for Text Classification," *arXiv:1509.01626 [cs]*, sep 2015.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.