Advanced Text Document Classification Techniques: Enhancing Accuracy and Efficiency in Real-World Applications

Bachelorarbeit

zur Erlangung des akademischen Grades

Bachelor of Science in Engineering (B.Sc.)

Eingereicht bei:

Fachhochschule Kufstein Tirol Bildungs GmbH

Web Business & Technology

Verfasser/in:

Jesse Lang

2110653142

Gutachter : Prof. (FH) Dr. Michael Kohlegger

Abgabedatum:

19. May 2024

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Bachelorarbeit selbständig und ohne fremde Hilfe verfasst und in der Bearbeitung und Abfassung keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe. Die vorliegende Bachelorarbeit wurde noch nicht anderweitig für Prüfungszwecke vorgelegt.

Kufstein, 19. May 2024

Jesse Lang

Contents

List of Figures

List of Listings

List of Acronyms

NLP Natural Language Processing

ML Machine Learning

DL Deep Learning

TF-IDF Term Frequency-Inverse Document Frequency

BoW Bag-of-Words

SVM Support Vector Machine

KNN K-Nearest-Neighbour

RF Random Forest

DT Decision Tree

NB Naïve Bayes

XGBoost Extreme Gradient Boosting

LR Logistic Regression

GPT Generative Pre-trained Transformer

BERT Bidirectional Encoder Representations from Transformers

FH Kufstein Tirol
Web Business & Technology

Abstract of the thesis: **Advanced Text Document Classification Techniques: Enhancing Accuracy and Efficiency in Real-World Applications**

Author: Jesse Lang
First reviewer: Prof. (FH) Dr. Michael Kohlegger

This thesis delves into the realm of advanced text document classification techniques, merging machine learning with natural language processing (NLP) to refine the accuracy and efficiency of categorizing diverse textual documents. The overarching goal is to discern the efficacy of such techniques in real-world applications and to unravel the factors influencing their performance.

Methodologically, the study embarks on a comprehensive journey. Beginning with data collection, it progresses through a series of text preprocessing steps, including tokenization, lowercase conversion, and stopwords removal, aimed at refining the data. Feature extraction techniques such as Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) are then employed. The data undergoes a rigorous train-test-split process to facilitate model evaluation. Selection and training of classification models, spanning traditional machine learning to deep learning approaches, follow suit.

The culmination of this research yields compelling insights. Findings underscore the potency of advanced text document classification techniques in heightening accuracy and efficiency. Through meticulous experimentation and analysis, critical revelations emerge regarding model performance, data preprocessing intricacies, and runtime considerations. Noteworthy trends and discoveries pertaining to model performance and data characteristics surface, illuminating the landscape of text document classification.

The implications of these findings resonate deeply within the field. By shedding light on the underlying dynamics of model performance and the synergy between machine learning and NLP, this study propels the discourse surrounding classification methods in practical contexts. Such insights hold promise for informing the development of more robust classification systems and strategies.

In conclusion, this thesis underscores the pivotal role of advanced text document classification techniques and their potential to revolutionize classification endeavors. Looking ahead, avenues for future exploration include delving into additional feature extraction methods, integrating domain-specific knowledge, and exploring ensemble learning approaches. Moreover, extending the application of these techniques across diverse domains and delving into interpretability and fairness considerations beckon as fertile grounds for future research.

19. May 2024

FH Kufstein Tirol
Web Business & Technology

Kurzfassung der Bachelorarbeit: **Advanced Text Document Classification Techniques: Enhancing Accuracy and Efficiency in Real-World Applications**

Verfasser: Jesse Lang
Gutachter: Prof. (FH) Dr. Michael Kohlegger

Diese Arbeit widmet sich fortgeschrittenen Techniken der Textdokumentklassifizierung, die maschinelles Lernen mit natürlicher Sprachverarbeitung (NLP) kombinieren, um die Genauigkeit und Effizienz bei der Kategorisierung verschiedener Textdokumente zu verbessern. Das übergeordnete Ziel besteht darin, die Wirksamkeit solcher Techniken in realen Anwendungen zu ergründen und die Faktoren zu entschlüsseln, die ihre Leistung beeinflussen.

Methodisch begibt sich die Studie auf eine umfassende Reise. Angefangen bei der Datensammlung durchläuft sie eine Reihe von Schritten zur Textvorverarbeitung, darunter Tokenisierung, Umwandlung in Kleinbuchstaben und Entfernung von Stoppwörtern, mit dem Ziel, die Daten zu verfeinern. Techniken zur Merkmalsextraktion wie Bag-of-Words (BoW) und Term Frequency-Inverse Document Frequency (TF-IDF) werden dann angewendet. Die Daten durchlaufen einen gründlichen Trainings-Test-Split-Prozess, um die Modellbewertung zu ermöglichen. Die Auswahl und Schulung von Klassifikationsmodellen, die von traditionellem maschinellem Lernen bis hin zu Ansätzen des tiefen Lernens reichen, folgt.

Das Ergebnis dieser Forschung liefert überzeugende Erkenntnisse. Die Ergebnisse unterstreichen die Wirksamkeit fortgeschrittener Techniken der Textdokumentklassifizierung bei der Steigerung von Genauigkeit und Effizienz. Durch sorgfältige Experimente und Analysen treten wichtige Erkenntnisse hervor, die sich auf die Leistung der Modelle, die Feinheiten der Daten-Vorverarbeitung

und Laufzeitüberlegungen beziehen. Bemerkenswerte Trends und Entdeckungen zu Modellleistung und Datencharakteristika kommen ans Licht und beleuchten das Gebiet der Textdokumentklassifizierung.

Die Implikationen dieser Ergebnisse hallen tief im Feld wider. Indem sie das Verständnis der zugrunde liegenden Dynamiken der Modellleistung und der Synergie zwischen maschinellem Lernen und NLP erhellen, treibt diese Studie den Diskurs über Klassifikationsmethoden in praktischen Kontexten voran. Solche Erkenntnisse versprechen, die Entwicklung robusterer Klassifikationssysteme und -strategien zu informieren.

Zusammenfassend betont diese Arbeit die entscheidende Rolle fortgeschrittener Techniken der Textdokumentklassifizierung und ihr Potenzial, Klassifikationsbemühungen zu revolutionieren. Ausblickend auf die Zukunft umfassen mögliche Forschungsrichtungen die Untersuchung zusätzlicher Techniken zur Merkmalsextraktion, die Integration domänenspezifischen Wissens und die Erkundung von Ensemble-Lernansätzen. Darüber hinaus laden die Erweiterung der Anwendung dieser Techniken auf verschiedene Domänen und die Untersuchung von Interpretierbarkeits- und Fairness-Aspekten als fruchtbare Gebiete für zukünftige Forschung ein.

19. May 2024

1. Introduction

The classification of text documents is essential across various applications. It demands a high accuracy and efficiency, in order to be beneficial. Traditional approaches often struggle with the diversity and complexity of modern textual data, emphasizing the need for advanced techniques. Leveraging Natural Language Processing (NLP), data mining, and Machine Learning (ML) techniques hold promise in overcoming these challenges by enabling automated categorization of textual documents based on their content, context, and semantics. These procedures contain numerous obstacles, including defining accurate annotation of documents, the implementation of dimensionality reduction techniques to address complexities, and utilizing appropriate classifier functions to obtain robust generalization while avoiding overfitting (Aurangzeb et al., 2010).

One of the main sources for textual documents these days is the internet, often known as the World Wide Web, the amount that is available to us is constantly increasing. Unstructured textual formats, such as reports, emails, opinions, and news stories, are thought to contain approximately 80% or more of an organization's information. Studies indicate that unstructured formats contain almost 90% of the world's data. There's a clear necessity for the automatic extraction of valuable insights from vast amounts of textual data to aid human analysis (Aurangzeb et al., 2010).

This study aims to investigate how these advanced classification techniques,

can enhance the accuracy and efficiency of categorizing diverse textual documents. Libraries, such as scikit-learn, NLTK, and Keras, and their broad spectrum of algorithms will be used to implement different approaches and explore the interactions between these techniques. With the use of a BBC text document dataset with a ".csv" format, the research seeks to gain insights into optimizing classification performance in real-world applications.

The following research question arises from the objective: "How can advanced text document classification techniques, incorporating machine learning and natural language processing, be effectively employed to improve the accuracy and efficiency of categorizing diverse textual documents, and what factors influence the performance of such classification models in real-world applications?"

1.1 Motivation

The current state of research in text document classification techniques, with ML and NLP, showcases significant progress toward enhancing efficiency and accuracy. Recent developments have seen the emergence of sophisticated deep learning architectures, such as Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN), which excel in processing complex textual data. Furthermore, has the integration of pre-trained language models like Generative Pre-trained Transformer (GPT) and Bidirectional Encoder Representations from Transformers (BERT) revolutionized feature representation, allowing models to capture difficult semantic nuances. This dynamic landscape reflects ongoing efforts to refine classification models, making them increasingly adept at real-world applications.

2. Theory

Through an insight into relevant development in the field, various models, preprocessing methodologies, and evaluation metrics, this chapter aims to construct a theoretical framework for the thesis. Its objective is to empower the reader with the requisite understanding to contextualize the research outcomes and their significance.

2.1 Natural Language Processing

Natural Language Processing (NLP) is a branch of artificial intelligence and computer science, that deals with the interaction between computers and human language (Helland, 2023). It aims to enable machines to generate human language, process, and understand it. By employing a variety of techniques and different approaches, such as deep learning, rule-based systems, and statistical methods, is NLP capable of tackling different language-related tasks. The usage of it can be found in numerous areas, including machine translation, chatbots, text classification, and speech recognition (Helland, 2023).

2.2 Machine Learning

Machine learning refers to the development of computer programs that are learning from experience to complete and solve a certain task. The measurement of the performance is calculated by the ability to do so (Helland, 2023). A training dataset represents the "experience" that is acquired by machine learning models, which contains output and input pairs. From the analysis of these examples can the model recognize and generalize the patterns to new, unseen data (Helland, 2023). To simplify, can it be reflected in the process of human learning and adaptation to new scenarios. Common applications include fraud detection, self-driving cars, and personalized recommendations (Helland, 2023).

2.3 Deep Learning

Deep learning is a method that uses non-linear modules to transform the data at multiple levels of abstraction, allowing models to find patterns in the raw data (Helland, 2023). This suggests that deep learning models are discovering and learning different data features without the need for human interaction (Helland, 2023). Because of its universal learning, generalization potential, robustness, and scalability advantages, this can be used in a variety of applications without the need for precise feature engineering (Helland, 2023).

2.4 Evaluation Metrics

A machine learning model's performance is assessed using metrics. They are employed to evaluate the accuracy of the made predictions, to analyze the output of various models, and to fine-tune them for optimal performance

(Helland, 2023). Different types of machine learning issues have various types of metrics available. The model selection process, the optimization procedure, and the overall understanding of the model's capabilities can all be impacted by the metrics chosen. Selecting the incorrect metrics might also result in a biased model, which aligns differently with the project's objectives (Helland, 2023).

2.4.1 Accuracy

One popular assessment metric for classification problems is "accuracy". Out of all the samples in the prediction, it calculates the proportion of correctly classified samples (Helland, 2023).

Accuracy = Number of correctly classified instances / Total number of instances

For balanced datasets, accuracy works well since it presents a realistic picture of the model's capabilities and performance (Helland, 2023). When datasets are unbalanced, accuracy might be misleading and more difficult to interpret (Helland, 2023). This may be due to a single label in the dataset that accounts for the majority of the samples; therefore, reasonable accuracy can still be obtained by projecting all samples to the dominant label. This does not imply that the model is good because it ignores the less common but no less significant labels. Accuracy in multi-label classification only takes into account samples where every label is correctly classified. Because of this, using accuracy as a multi-label classification metric to evaluate the performance of multi-label models is more strict, less informative, and less desirable (Helland, 2023).

3. Methodology

The methods used to address the research topics raised in this thesis are described in this chapter. To obtain the results that will be presented, this section provides a thorough overview of the pipeline (Figure 1), outlining the procedures for preprocessing data, extracting features, model selection and training, and assessing performance. Through the conversion of theoretical ideas into practical procedures, aiming to offer an outline of how these methods are applied in real-life situations.

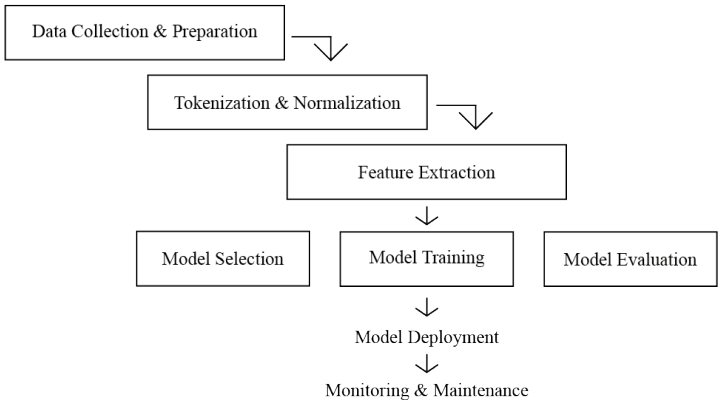

Figure 1: Implementation pipeline

3.1 Data Collection

The collection of data is a crucial step that lays the foundation for later analysis and model development. It involves gathering textual data from various

sources, which could include websites, databases, or specialized datasets for the specific domain of interest. It is important to collect a sufficiently diverse amount of data to capture the variability present in real-world text data (GPT-3, 2024). This thesis uses a BBC news dataset containing 2225 text data and five categories of documents (Thite, 2024).

3.2 Data and Text preprocessing

Preprocessing methods are an essential step for text mining techniques and applications. The data and its columns need to be analyzed and inspected since it is often necessary to generate a new column combining the various features. Through the joint column, a better, more comprehensive, and more accurate analysis can take place. The three essential preprocessing steps — tokenization, lowercase, stop word removal, and TF/IDF algorithms — are covered in this study (Figure 1).

3.2.1 Tokenization

Tokenization is the process of splitting sentences into individual words, characters, and punctuation, which are referred to as tokens (Tabassum and Patil, 2020). The split function uses white spaces or punctuations as dividing criteria. These generated tokens are often stored in a list afterward. In later processing phases, this step aids in removing unnecessary terms (Tabassum and Patil, 2020).

For example:

“This is an example sentence for the showcase of tokenization!”

Will be split into:

“This”, “is”, “an”, “example”, “sentence”, “for”, “the”, “showcase”, “of”, “tokenization”, “!”

3.2.2 Lowercase Conversion

Text typically consists of capital letters and abbreviations. Although this stage of text preprocessing is frequently skipped, is it one of the easiest and most successful ones (Tabassum and Patil, 2020). NLP is case-sensitive, meaning, it interprets ’Hello’ differently than ’hello’ and leads to a different outcome. In the later phases of word embedding would it create two distinct vectors, for the same words with one in capital and one in lowercase. For this reason, the best practice in text pre-processing has been to make all words lowercase (Tabassum and Patil, 2020).

3.2.3 StopWords Removal

Simple words like "the", "are", "is", "and" and so forth have no significance except in certain particular use cases. For instance, these extra words are not given any weightage in the text classification use case (Tabassum and Patil, 2020). The keywords that define the topics are the only ones that are extracted. Therefore, to reach the best result with the algorithms these StopWords have to be found and removed from their document. It is also important to remember that in some scenarios, such as conversational models, the inclusion of specific negation words, like “No”, “cannot”, “wont” and “not”, is crucial (Tabassum and Patil, 2020). Libraries like NLTK and sklearn already offer predefined lists of StopWords which can be easily downloaded and implemented into the code.

3.3 Feature Extraction

Feature extraction involves converting features into vector representations for machine comprehension tasks. Following extraction, each feature is transformed into a vector format before being fed into classifier models. Common techniques like Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) are frequently employed for this purpose and will be explored further.

3.3.1 Bag-of-Words (BoW)

A Bag-of-Words in terms of natural language processing is a collection of words based on the number of occurrences in a given text or document. It only counts the frequency of a word, regardless of its position in the text (Tabassum and Patil, 2020). Thus, the BoW interprets that documents or texts containing similar words share the same context. One flaw of the model is that it prioritizes words that appear more frequently, making them more significant. On the other hand, some words may occur more frequently than others but lack sufficient information to help in clustering or classification issues. Additionally, longer documents provide a greater rate than shorter ones, which reduces the accuracy of the BoW model (Tabassum and Patil, 2020). Example text (GPT-3, 2024):

```
Document 0: "The quick brown fox"
Document 1: "Jumped over the lazy dog"
Document 2: "The dog chased the fox"

Vocabulary:
{'the': 8, 'quick': 7, 'brown': 0, 'fox': 3, 'jumped': 4,
'over': 6, 'lazy': 5, 'dog': 2, 'chased': 1}
```

```
BoW:
    brown  chased  dog  fox  jumped  lazy  over  quick  the
0       1       0    0    1       0     0     0      1     1
1       0       0    1    0       1     1     1      0     1
2       0       1    1    1       0     0     0      0     2
```

3.3.2 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF, or Term Frequency-Inverse Document Frequency, serves as a numerical metric to gauge the significance of a word within a document collection. Widely applied in information retrieval and text mining, TF-IDF assigns weight to words based on their frequency in a document relative to their occurrence across the corpus (Vijayarani et al., 2015). This weighting mechanism effectively mitigates the influence of commonly used words. Employing TF-IDF for StopWords filtering proves beneficial across various domains, including text classification and summarization. The model combines two key metrics: term frequency and inverse document frequency, wherein the former tallies the occurrences of each term in documents, while the latter adjusts for the term's prevalence across the corpus (Vijayarani et al., 2015). By taking the logarithm of the ratio of all documents to the instances of a term within a document, TF-IDF effectively diminishes the importance of less significant words (Tabassum and Patil, 2020). Example text (GPT-3, 2024):

```
Document 0: "The quick brown fox"
Document 1: "Jumped over the lazy dog"
Document 2: "The dog chased the fox"

Vocabulary:
{'the': 8, 'quick': 7, 'brown': 0, 'fox': 3, 'jumped': 4,
'over': 6, 'lazy': 5, 'dog': 2, 'chased': 1}
```

```
TF-IDF:
   brown  chased   dog   fox   jumped  lazy   over  quick   the
0   0.58    0.00  0.00  0.44    0.00  0.00   0.00   0.58   0.35
1   0.00    0.00  0.38  0.00    0.50  0.50   0.50   0.00   0.30
2   0.00    0.53  0.40  0.40    0.00  0.00   0.00   0.00   0.63
```

3.4 Train-Test-Split

Train-Test-Split is a common method provided by sklearn, used to divide a given dataset into smaller subsets. The datasets need to be divided into training and testing sets in order to fairly assess the models. By dividing the data, the model's performance on unknown data may be evaluated, revealing whether the model overfits or performs well in terms of generalization. This stage is crucial because the model's ultimate objective is to accurately predict new data.

Every dataset was split only once, and all models were trained and fine-tuned using the same split, to maintain the comparison as fair and repeatable as possible. Listing 1 displays the implementation, which divides the data into a split of 80:20, where the training set contains 80%, and the testing set 20%.

Listing 1: Train-Test-Split in Python (Scikit-learn, 2024b)

```
1 from sklearn.model_selection import train_test_split
2
3 X_train,X_test,y_train,y_test = train_test_split(X,y,
    test_size=0.2, random_state=42)
```

The variable X (features) is assigned to the previously created vector and y (target) to the labels that should be predicted.

The random_state parameter controls the randomness of the data splitting pro-

cess. When a specific random_state value is provided, the data splitting process will produce the same result every time it is executed, ensuring reproducibility. If random_state is not specified, the data splitting will be different each time the function is called.

3.5 Model Selection

Machine Learning is split into three main categories, which are supervised, unsupervised and reinforcement learning. This thesis only required supervised learning, which is about the prediction of values with regression models, as well as classifying data with predefined labels. On the other hand, there is unsupervised learning which contains the analysis of patterns and can form clusters out of unlabelled data.

3.5.1 Classification Models

There are several different classification models and each of them fits a specific use case best. The models need to be evaluated and compared to one and another, to find the optimal algorithm. This study analyses seven different models from sklearn (Naive Bayes, Random Forest, Support Vector Machine, KNeighborsClassifier, Logistic Regression, Decision Tree, XGBoost) and evaluates them based on the run time and accuracy.

K-nearest neighbour - KNN

The K-Nearest-Neighbour (KNN) algorithm is one of the finest examples of instance-based learning (Sen et al., 2020). Additionally, it is easy to understand and a simple method for classification problems. Despite its simplicity, it

has the capability to yield results that are highly competitive. Not only is it well suitable for classifications but it also fits the requirements for regression predictions (Sen et al., 2020).

The algorithm stores all the given data points and predicts the target based on giving attention to the similarity measurements of the surrounding neighbours in likelihood (Sen et al., 2020). The number of neighbours that will be taken in consideration is defined by the "k" variable. Assuming k equals 3, a circular region with the new data point as its centroid is created to encompass only the three closest neighbouring data points on the plane. The determination of the label for the new data point is then based on the distances between the data point and each of its neighbours (Sen et al., 2020).

Some of the advantages are that it handles noisy and large training data well, besides the simplicity of the implementation (Sen et al., 2020). A significant limitation of this algorithm arises from the necessity to recalculate the distances from K neighbours for every new instance, resulting in substantial computational time consumption. Additionally, accurately determining the value of K is crucial to achieve a lower error rate (Sen et al., 2020).

Support Vector Machine - SVM

Another supervised algorithm is the Support Vector Machine (SVM). It can handle both, classification, and regression problems, though is it more seen for classification. Furthermore, it can manage numerous instances that involve both continuous and categorical data (Sen et al., 2020).

The algorithm can be defined like following. Items of the dataset with "n" features will be characterised and plotted as points in an n-dimensional space split into classes by a hyperplane with the widest possible margin. The data points are then mapped into the previous defined space to predict their label

based on their position relative to the hyperplane (Sen et al., 2020).

A significant performance boost can be seen, when the variable "n" exceeds the total size of sample set. Therefore, is this algorithm mostly taken under consideration for high-dimensional data (Sen et al., 2020). Further improvements in performance can be achieved by having a well-constructed hyperplane. Despite its advantages, is a relatively high training time one of its drawbacks. Which leads to slower predictions, especially with large datasets (Sen et al., 2020).

Decision Tree - DT

One kind of supervised learning method for regression and classification is the use of Decision Trees (DTs) (Scikit-learn, 2024a). The objective is to create a model that can forecast a target variable's value using basic decision rules deduced from the data attributes. A piecewise constant approximation can be thought of as a tree (Scikit-learn, 2024a). For instance, decision trees estimate a sine curve depending on inputs by combining a set of if-then-else decision rules. As the tree goes deeper, the model fits the data better and the decision criteria get increasingly complex (Scikit-learn, 2024a).

The ease of use and interpretability of decision trees is one of its main benefits. It is possible to visualise and comprehend the tree structure, so even non-experts may use it (Scikit-learn, 2024a). Furthermore, because decision trees can handle both numerical and categorical data without requiring a lot of preprocessing, they also require less preparation of the data. They also have the benefit of handling multi-output issues and offering a white box approach, in which a decision's logic may be simply described using boolean logic (Scikit-learn, 2024a).

Nevertheless, decision trees can overfit, especially if they get too complicated.

To avoid this problem, measures like trimming and imposing tree growth restrictions are required (Scikit-learn, 2024a). Decision trees can also be unstable since slight changes in the data might produce noticeably different tree architectures. Despite these drawbacks, is it a useful tool in machine learning, even with these limitations, especially where simplicity and interpretability are top priorities (Scikit-learn, 2024a).

3.5.2 Deep Learning

Deep Learning (DL) is a specific category within ML methodologies that utilizes Artificial Neural Networks (ANN). These networks are loosely inspired by the structure of neurons found in the human brain (Gulli and Pal, 2017). Informally, the term "deep" originally referred to the presence of numerous layers in the artificial neural network. However, this definition has evolved over time. While just four years ago, having 10 layers was considered sufficient to qualify a network as deep, today it is more commonplace to characterize a network as deep when it comprises hundreds of layers (Gulli and Pal, 2017).

Keras serves as a user-friendly high-level deep learning library in Python, providing a convenient interface for building neural networks (Gulli and Pal, 2017). The sequential model in Keras is a linear stack of layers, allowing the straightforward construction of neural networks by sequentially adding layers. Each layer, often employing activation functions like Rectified Linear Unit (ReLU), introduces non-linearity to the model, enabling it to capture complex patterns and relationships within the data (Gulli and Pal, 2017).

4. Results

The results presented in this chapter are the outcomes obtained by applying pre-processing techniques and machine learning models to the dataset for multi-class classification. Chapter 5 goes into further depth about the findings. Each of the model's runtimes, performance metrics (accuracy), and findings of the data preprocessing steps are included in the results.

4.1 Data Preprocessing

4.1.1 Label Distribution

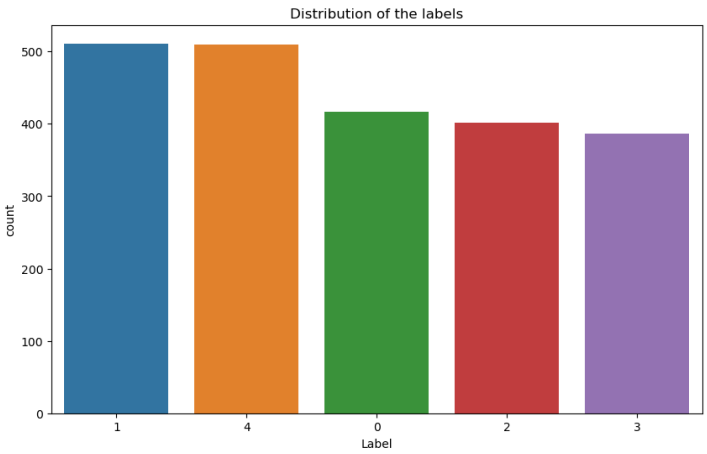

Figure 2: Distributon of the labels

Figure 2 shows the distribution of the labels in the BBC dataset. The labels

represent the categories 0: politics, 1: sport, 2: technology, 3: entertainment, 4: business. The distribution is measured in the total amount of documents per label.

As can be seen in Figure 2, there is an unequal distribution of the categories, where the sports and business sections carry the most weight. Whereas the other three labels have around 20% less data available.

4.1.2 Document length distribution

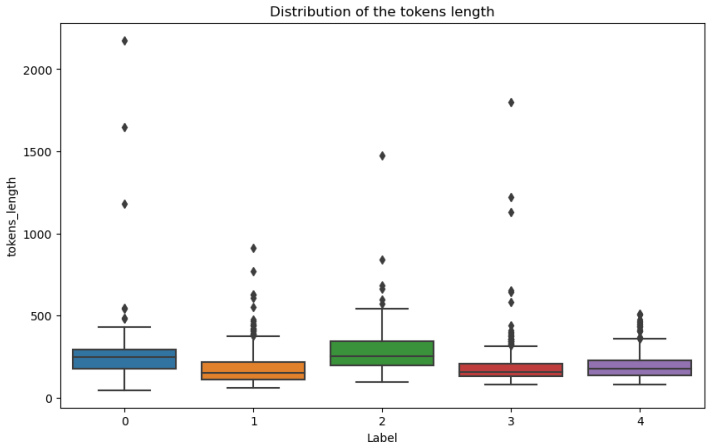

Figure 3: Distributon of the document length

Figure 3 illustrates the distribution of document lengths per label within the dataset using box plots with outliers. Each box plot represents the distribution of document lengths, with the box indicating the interquartile range and the median, while the outer ends extend to the minimum and maximum values.

This visualization provides a comprehensive view of the variability in document lengths present in the textual data. By examining the box plots, we can discern the typical range of token lengths as well as the presence of outliers or extreme values.

4.1.3 Label insights

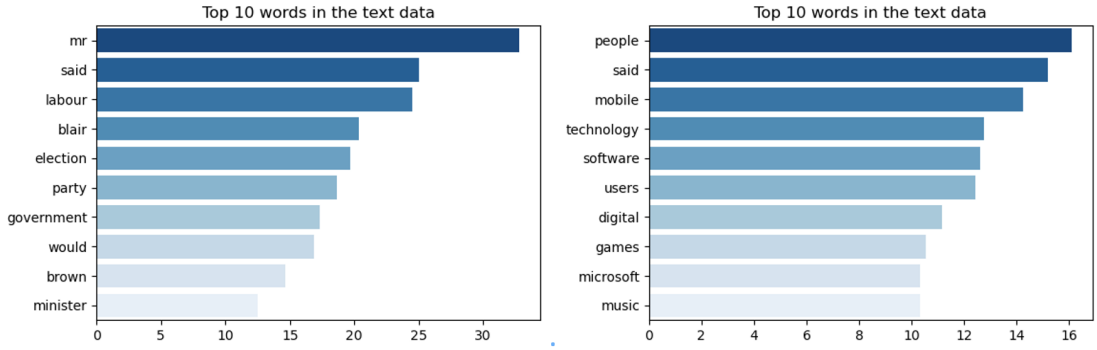

Figure 4: Top 10 words of the politics and technology category

Figure 4 presents the top 10 words for each of the two labels, "politics" and "technology." This visualization offers a concise representation of the most frequent words associated with each category, providing a quick overview of the predominant themes within the dataset.

By presenting the most commonly occurring words, this figure aids in understanding the lexical characteristics of the politics and technology categories, serving as a useful reference for subsequent analysis and interpretation of the textual data.

4.2 Model Performance

By analyzing key metrics like accuracy and runtime, it provides important insights into each model's effectiveness and computational efficiency. This evaluation process guides our decisions in selecting the most suitable models for practical applications.

4.2.1 Accuracy

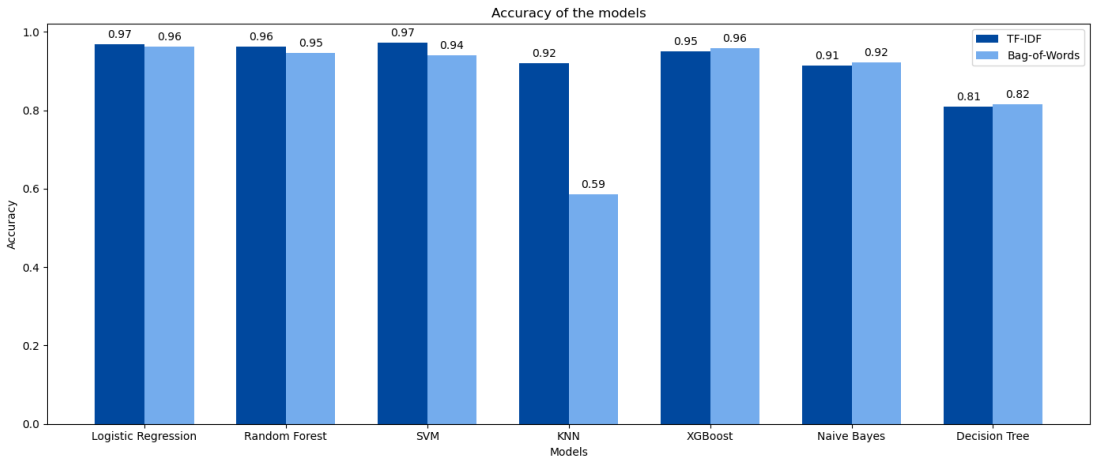

Figure 5: Comparison of model accuracy with TF-IDF and BoW

Figure 5 displays the evaluation of different classification models with two approaches, TF-IDF and BoW. In this visualization, each model's accuracy score is shown, providing a clear comparison of their performance in classifying textual documents across different categories or classes. Looking at the TF-IDF approach, Logistic Regression and Support Vector Machine provided the most precise score with 0.97 accuracy, followed by Random Forest (0.96) and K-Nearest-Neighbour (0.92). In contrast, the Decision Tree model came out as the least accurate model (0.81).

This information is crucial for determining the effectiveness and reliability of the classification algorithms employed in the study, aiding in model selection and optimization strategies.

4.2.2 Runtime

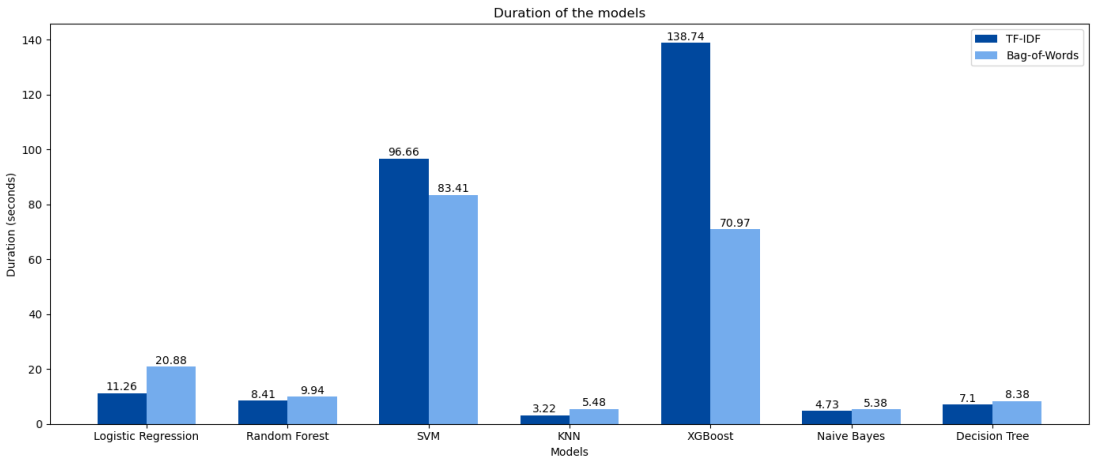

Figure 6: Comparison of model runtime with TF-IDF and BoW

Figure 6 displays the runtime of the classification models (TF-IDF and BoW approach) utilized in this study. Runtime refers to the time taken by each model to complete the classification task, from training to prediction, on the given dataset. The faster the model's runtime is the more efficient it will be in production. Therefore, considering the TF-IDF approach, is KNN (3.22 s) the quickest, Naïve Bayes (4.73 s) second, and Decision Tree (7.1 s) third. At the bottom of the ranking are the following models, SVM (96.66 s) and XGBoost (138.74 s).

This information is important for assessing the practical probability and scalability of the classification algorithms, particularly in real-world applications where computational efficiency is a significant consideration.

5. Discussion

In addition to interpreting the machine learning experiment findings, the discussion chapter will compare the outcomes of the various techniques by looking at the evaluation metrics and exchanging opinions regarding the models, data, and decisions made in this thesis. General questions about NLP and the potential applications of text document classification will also examined.

5.1 Analysis of the Results

5.1.1 Model Performance

Figure 5 and Figure 6 showcase the performances of the machine learning models with two different feature extraction approaches.

When employing the TF-IDF approach, several models demonstrated strong performance. Logistic Regression achieved one of the highest accuracy among all models, with an impressive 96.86%. Its relatively low duration indicates efficient training and inference times, making it suitable for real-time applications. Random Forest also exhibited competitive performance with an accuracy of 96.26% and a shorter duration compared to Logistic Regression, suggesting its efficiency in classification tasks. However, the Support Vector Machine (SVM) model outperformed others in accuracy, achieving 97.16%. Nonetheless, its

longer duration implies higher computational complexity, which may limit its scalability for large datasets or real-time applications. Despite its simplicity, K-Nearest Neighbors (KNN) achieved a respectable accuracy of 91.92% with the shortest duration among all models, indicating its efficiency for tasks with computational constraints. XGBoost, although offering high accuracy (95.06%), required the longest duration, highlighting a trade-off between accuracy and computational cost. Naive Bayes demonstrated efficient performance with a balanced accuracy (91.32%) and a short duration, making it a viable choice for resource-constrained environments. Conversely, the Decision Tree model exhibited the lowest accuracy (80.84%) with moderate duration, suggesting limited performance compared to other models.

Transitioning to the Bag-of-Words (BoW) approach, models maintained competitive accuracy levels, though with slight variations in performance. Logistic Regression and Random Forest retained strong performance, achieving accuracies of 96.26% and 94.61%, respectively, comparable to their TF-IDF counterparts. However, the computational overhead increased slightly, indicating a trade-off between accuracy and computational efficiency with BoW representation. SVM demonstrated a similar trend, achieving an accuracy of 94.01% with reduced computational complexity compared to TF-IDF. KNN experienced a significant decrease in accuracy (58.53%) with BoW representation, suggesting its limitations in handling sparse feature spaces. XGBoost continued to perform well with BoW, achieving an accuracy of 95.81%, albeit with reduced computational time compared to TF-IDF. Naive Bayes and Decision Tree models maintained consistent performance with BoW, offering high accuracy levels (92.22% and 81.59%, respectively) and efficient computational times.

In summary, the choice of feature representation and model selection depends on the specific requirements of the text classification task. TF-IDF provides discriminative features that may enhance model performance, while BoW offers computational efficiency, particularly for resource-constrained environments.

Logistic Regression and Random Forest models present a favorable balance between accuracy and computational cost, making them suitable choices for various applications. Future research directions may involve exploring ensemble methods or deep learning approaches to further enhance classification accuracy while maintaining computational efficiency. Additionally, hyperparameter tuning and feature engineering techniques could provide insights into improving model performance, particularly for Decision Trees and KNN with BoW representation.

5.1.2 Data Pre-processing

Figure 2 and Figure 3 demonstrate two important distributions within the dataset. An uneven amount of labels (Figure 2) can disrupt the performance and accuracy of the model. In this example, there would be more weightage on the labels 1 and 4. While splitting the data into training and testing parts, can the "stratify" parameter be used to help balance the two sets for an even more distribution and leads to more accurate results. Figure 3 indicates that there are documents with various amounts of length. Longer documents will help the model gain more insight into the category, whereas shorter documents with only a few tokens can lead to confusion since they will probably not contain any significant words about the topic.

While exploring the feature representations (either in BoW or TF-IDF format) is it helpful to use topic modelling (Figure 4) for a deeper comprehension of the different categories. Due to this, can major topics already be identified and annotated. On the left side in Figure 4 is the topic visualization of the label "politics". Most appeared words are mr, labour, election, party, government, minister, which are words defining politics. The right sight illustrates the label "technology", seen by the tokens such as mobile, technology, software, digital. After this analysis is the data more clear and easier to understand. Annotating

the data for classification tasks is a crucial step when working with a dataset with no predefined labels.

5.2 Conclusion

This thesis' main objective was to investigate advanced text document classification techniques integrating machine learning and natural language processing to enhance the accuracy and efficiency of categorizing diverse textual documents. Our research question aimed to understand how these techniques could be effectively employed in real-world applications and identify factors influencing their performance.

Through comprehensive experimentation and analysis, there have several key observations been made regarding the effectiveness of different classification methods. Logistic Regression and Random Forest models consistently demonstrated strong performance across various feature representations, showcasing their versatility and robustness. These models offered competitive accuracy levels while maintaining computational efficiency, making them practical choices for text classification tasks in diverse domains.

Furthermore, the study highlighted the importance of feature representation methods, such as TF-IDF and Bag-of-Words, in influencing classification performance. While TF-IDF captures the importance of terms within documents, BoW provides computational efficiency, particularly in resource-constrained environments. Understanding the trade-offs between these representations is crucial for selecting appropriate techniques based on specific task requirements.

Challenges and limitations in certain classification approaches have also been identified. Support Vector Machine (SVM), while achieving high accuracy, exhibited longer training and inference times, limiting its suitability for real-time applications. Additionally, K-Nearest Neighbors (KNN) showed efficiency in

computational time but experienced a decrease in accuracy with sparse feature spaces, suggesting the need for alternative approaches in such scenarios.

Limitations of this research include the use of a single dataset and limited hyperparameter tuning, which may impact the generalizability of the findings. Future studies could explore the performance of classification models across multiple datasets and conduct extensive hyperparameter optimization to enhance model performance. Additionally, investigating the impact of feature engineering techniques and ensemble methods on classification accuracy could provide valuable insights into improving model robustness and efficiency.

In conclusion, the research contributes to the advancement of text document classification methodologies by providing insights into the performance of advanced techniques in real-world settings. By understanding the factors influencing classification model performance, we can better guide the development and deployment of text classification systems across various domains. Future research directions may involve exploring ensemble methods, deep learning approaches, and fine-tuning techniques to further enhance classification accuracy while optimizing computational efficiency.

Ultimately, the findings underscore the importance of leveraging advanced techniques in text document classification to meet the evolving demands of information processing and analysis in today's data-driven world.

Bibliography

Aurangzeb, K., Baharum, B., Lam Hong, L., and Khairullah, K. (2010). A Review of Machine Learning Algorithms for Text-Documents Classification. *Journal of Advances in Information Technology*, 1(1):1–1.

GPT-3, O. (2024). Chat gpt-3. https://chat.openai.com. Accessed: 2024-02-18.

Gulli, A. and Pal, S. (2017). *Deep learning with Keras*. Packt Publishing Ltd.

Helland, E. D. (2023). *Tackling Lower-Resource Language Challenges: A Comparative Study of Norwegian Pre-Trained BERT Models and Traditional Approaches for Football Article Paragraph Classification*. PhD thesis.

Scikit-learn (2024a). Decision trees. https://scikit-learn/stable/modules/tree.html. Accessed: 2024-02-22.

Scikit-learn (2024b). Train-test split. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html. Accessed: 2024-03-14.

Sen, P. C., Hajra, M., and Ghosh, M. (2020). Supervised classification algorithms in machine learning: A survey and review. In *Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018*, pages 99–111. Springer.

Tabassum, A. and Patil, R. R. (2020). A survey on text pre-processing & feature extraction techniques in natural language processing. *International Research Journal of Engineering and Technology (IRJET)*, 7(06):4864–4867.

Thite, S. (2024). Text document classification dataset. https://www.kaggle.com/datasets/sunilthite/text-document-classification-dataset. Accessed: 2024-02-18.

Vijayarani, S., Ilamathi, M. J., Nithya, M., and others (2015). Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks*, 5(1):7–16.

A. Github Repository

The code for this thesis is available on GitHub. The repository contains the code for the data preprocessing, the model training, and the evaluation of the model. The code is written in Python and uses the libraries TensorFlow, Keras, and Scikit-learn. The repository can be found at the following link:

https://github.com/jlang62/bachelor-thesis/tree/main