EXPLORING SENTIMENT CLASSIFICATION : A COMPARATIVE ANALYSIS OF DEEP LEARNING AND MACHINE LEARNING METHODS

Noureddine BERTRAND, Dorian GROUTEAU, Olivier JIANG, Lukas KAIBEL, Hui ZHAO

UFR Mathématiques et Informatique, Université of Paris cité

Mathematik und Informatik, Freie Universität Berlin

E–mail: {*noureddine-kamil.bertrand — dorian.grouteau — olivier.jang — hui.zhao*}*@etu.u-paris.fr, lukas.kaibel@fu-berlin.de*

SUMMARY: The aim of this paper is to compare different classification methods with the objective of analyzing sentiments through textual data. We compare classical machine learning classification methods with more recent deep learning methods. Our goal is to provide insights into the performance, scalability, and interpretability of these methods, thus guiding practitioners in choosing the most suitable approach for sentiment classification tasks.

Key words. Machine Learning - Deep Learning - Data - Sentiment analysis - Textual data - Scalability

1. INTRODUCTION

Today, with a continuously growing amount of data, the ability to differentiate sentiments from vast volumes of data has become paramount. Understanding the sentiments expressed in text, particularly in the form of user reviews, social media posts, or customer comments, is of utmost importance for businesses and researchers, for example. Sentiment classification into positive, negative, or neutral categories not only helps assess public opinion but also facilitates decision-making processes in various domains.

Traditionally, sentiment analysis has relied on conventional machine learning techniques such as Support Vector Machines (SVM), Naive Bayes, or logistic regression, which often require manually crafted features and struggle to capture complex linguistic patterns. However, the emergence of deep learning has revolutionized sentiment analysis by enabling models to automatically learn complex representations of textual data, thereby surpassing the limitations of traditional methods.

In this report, we focus on the domain of sentiment classification, comparing deep learning methods to traditional machine learning approaches. We will explore their effectiveness in distinguishing between positive and negative sentiments from textual data, aiming to highlight the strengths and weaknesses of each approach. Through conducting an in-depth analysis, we aim to provide insights into the performance, scalability, and interpretability of these methods, thus guiding practitioners in choosing the most suitable approach for sentiment classification tasks.

We also used the TextBlob library to compare the results obtained with traditional classification methods and deep learning. TextBlob is a popular Python library that provides tools for natural language processing (NLP).

It is also worth noting that our study will be based on a variety of representative datasets, covering different domains and types of text. This approach will allow us to assess the generalization of sentiment classification methods across various contexts and better understand their robustness against data variability. By integrating a diversity of corpora, we can examine how model performances vary depending on the specific nature of textual data, enriching our comparative analysis and providing more informed recommendations to practitioners. The remainder of the report is structured as follows: Section 2 provides an overview of related work in sentiment analysis. Section 3 covers the dataset and the database used. Section 4 outlines the methods used and in Section 5 their implementation. Section 6 discusses the results.Finally, Section 8 presents our conclusions and suggests avenues for further improvement.

2. RELATED WORKS

Understanding the landscape of sentiment analysis involves a deep dive into the methodologies and techniques employed across both traditional machine learning and deep learning domains. This section provides an overview of the related works in sentiment analysis research, encompassing various approaches, comparisons, challenges, and future directions.

2.1. Embedding

Word embeddings provide numerical representations for words and phrases in NLP. In Perti Ash-

win and Ankit (2023)'s study, N-gram-based word embeddings were used to create vector representations of tweets from Twitter for sentiment analysis. These vectors were then used in automated processes to classify the tweets as either positive or negative. They tried to test the performances of many models, one of them combining a CNN model with SVM. They replaced the last dense layer with SVM for classification. SVM has several advantages that can improve the performance of the model, including the ability to handle non-linear data, good regularization, and robustness to outliers. In a comparison of different models for sentiment analysis of online product reviews on Twitter, they showed that the hybrid model achieved the highest accuracy score. The study focused on N-gram-based word embedding, with plans to explore BERT-based embeddings in the future. Erkan and Gungor (2023) compares various embeddings and tokenization mechanism for English and Turkish comment. They compare the combination of Word2Vec and GloVe embeddings with pretrained embedding models, namely, BERT and RoBERTa, across CNN and FFNN models. This study emphasize the importance of the embedding and tokenization choice as well as it's various effect in English and Turkish language.

2.2. Traditional Machine Learning Approaches

Traditional machine learning methods, such as Support Vector Machines (SVM) and Naive Bayes, have been foundational in sentiment analysis tasks. These techniques typically rely on handcrafted features, such as bag-of-words or n-grams, combined with lexical resources like sentiment lexicons. While these methods have demonstrated effectiveness, they often struggle with capturing the nuanced and complex linguistic patterns present in natural language text.

2.3. Deep Learning Techniques

Deep learning has emerged as a powerful paradigm for sentiment analysis, leveraging techniques such as Recurrent Neural Networks (RNNs) Durga and Godavarthi (2023), Convolutional Neural Networks (CNNs) Erkan and Gungor, and transformer-based architectures like BERT and GPT. These models excel at automatically learning intricate representations of textual data, allowing them to capture complex contextual dependencies and achieve state-of-the-art performance on various sentiment analysis benchmarks. Deep learning approaches offer the potential for automatic feature learning, reducing the need for handcrafted features and improving adaptability to diverse datasets.

2.4. Performance Comparison

Numerous studies have compared the performance of traditional machine learning algorithms with deep learning models in sentiment classification tasks. While deep learning models often demonstrate superior performance, they also tend to require larger amounts of labeled data and computational resources for training. Additionally, understanding and interpreting the decisions made by deep learning models can be challenging due to their perceived opacity, leading to concerns about their trustworthiness and reliability in real-world applications.

Therefore, sometimes we need to use Multimodal of deep learning (pre-trained Inception network and GloVe word embeddings) to accurately infer the user's feelings when analyzing images and text in Tumblr posts Hu and Flaxman (2018).

2.5. Challenges and Limitations

Sentiment analysis faces several challenges, particularly with deep learning models known for their "black box" nature, which makes them difficult to interpret. This lack of transparency can lead to a lack of trust and hinders understanding, especially in sensitive fields like healthcare and legal domains. Addressing these issues requires the development of explainable AI methods that can reveal how deep learning models make decisions, providing clarity to users and stakeholders.

Another major challenge is the selection and preparation of datasets. With large amounts of data available, it is vital to properly label and curate it to meet specific requirements. Using diverse training techniques during the data labeling process can improve accuracy and reduce errors in later analyses. This careful approach to data preparation is key to optimizing the performance of machine learning models across different use cases.

The Tsentiment15 dataset, presented by Iosifidis and Ntoutsi (2017), exemplifies the importance of strategic data selection, allowing for both batch and stream processing. This study underlines the significance of adapting dataset criteria to maintain robustness and precision in sentiment analysis.

2.6. Emerging Trends and Future Directions

Despite the challenges, ongoing research efforts are exploring novel architectures and methodologies to improve the interpretability, scalability, and generalization of sentiment analysis models. Integrating explainable AI techniques with deep learning architectures holds promise for enhancing model transparency and interpretability. Furthermore, the exploration of multi-modal and multi-task learning approaches may enable sentiment analysis models to leverage additional sources of information, such as images or user demographics, to enhance performance and robustness.

By critically examining the advancements, challenges, and future directions in sentiment analysis research, this review provides insights into the current state of the field and guides future research endeavors towards addressing key challenges and unlocking new opportunities for innovation.

2.7. Bitcoin Sentiment Prediction

To illustrate a practical application of sentiment analysis, we reference Feizian and Amiri (2023) who utilize this technique to predict the Bitcoin market prices. This study, which draws from a dataset of Bitcoin-related tweets from Kaggle, likely aligns with the dataset we plan to use for our analysis due to similarities in entries and timeframe. The researchers initially cleaned the data by removing tweets with irrelevant hashtags and filtering out non-alphanumeric characters from the text.

For the sentiment analysis, they employed the TextBlob library to determine the sentiment of each tweet Loria (2023). This sentiment data was then correlated with the corresponding Bitcoin market prices on the same dates.

Furthermore, the study involved training a Long Short-Term Memory (LSTM) model to forecast Bitcoin prices based on the sentiment derived from public tweets and additional user metrics, such as follower count. By incorporating both sentiment data and user information, their model demonstrated superior performance compared to models lacking this information Feizian and Amiri (2023).

Our report will focus solely on replicating the sentiment analysis segment from this study to demonstrate a valuable application of this technique Feizian and Amiri (2023).

3. DATASET

As part of our comparative analysis of sentiment classification methods, the selection and characterization of datasets play a crucial role in ensuring the validity and generalizability of the study's results. Here, we present three distinct datasets from different domains, each providing a diverse set of textual data for sentiment analysis tasks. These datasets are stored and managed within a non-relational database system, enabling efficient storage, retrieval, and manipulation of data.

3.1. IMDB Movie Reviews Dataset

The IMDB movie reviews dataset comprises a collection of user reviews extracted from the IMDB website, a popular platform where movie enthusiasts can rate and review films. This dataset encompasses a wide variety of film genres, release years, and audience types, providing a rich source of textual data for sentiment analysis. Each review is associated with a corresponding rating (e.g., star rating or numerical score) provided by the user, enabling the labeling of sentiment polarity (positive/negative) based on the sentiment expressed in the review.

3.2. Amazon CD & Vinyl Reviews Dataset

The Amazon CD & Vinyl reviews dataset consist of more than 1,400,000 reviews. This dataset is unlabeled and skewed. We reduce the size of the dataset to 269586, 89862 for each classes. This make the dataset unskewed. We attribute the label on each of the review based on the 5 stars rating. Review with less than 3 stars are labeled as negative, 3 stars is neutral and more than 3 is positive. This processing may introduce noise or wrong labeling to the dataset but allow us to quickly gather a labeled dataset.

3.3. Bitcoin Tweets Dataset

For the sentiment analysis of Bitcoin tweets, we utilized the "Bitcoin Tweets" dataset by Suresh (2023). This dataset comprises over 600,000 English-language tweets from around the globe, each accompanied by relevant metadata. The metadata includes details such as the author's name, hashtags, posting date, text content, among others, totaling 13 distinct types of data per tweet. The tweets were selected using filters for the hashtags #btc and #Bitcoin within the Twitter database, covering a period from February 2021 to February 2023.

3.4. Utilization of Non Relational Database

In our comparative investigation into sentiment classification methodologies, effective dataset management plays a pivotal role in ensuring the dependability and coherence of our analyses. Given the unstructured nature and substantial (potentially growing) size of our dataset, along with its sole purpose of serving as a data source without necessitating intricate queries or joins, we have chosen to employ a relational database system. This decision is motivated by the array of benefits it offers for data manipulation and organization:

(i) Data Flexibility: Non-relational databases offer flexibility in data modeling, accommodating diverse data structures such as documents, key-value pairs, wide-column stores, and graph data. This flexibility allows for storing and managing various datasets without the need for a rigid schema, promoting adaptability and agility in data management.

(ii) Scalability: A key advantage of non-relational databases is their ability to scale horizontally to handle large volumes of data efficiently. This scalability enables working with datasets of varying sizes and dynamically scaling resources to meet changing demands, ensuring consistent performance even as data grows.

(iii) Simplified Querying: Non-relational databases provide simplified querying mechanisms tailored to specific data models. These databases offer efficient retrieval methods optimized for the data structure, allowing for fast and effective data analysis without the complexity of traditional SQL queries.
(iv) Schema Flexibility: Unlike relational databases, non-relational databases do not enforce a fixed schema, allowing for dynamic schema evolution and accommodating evolving data requirements. This schema flexibility simplifies data management tasks and supports agile development processes.
(v) Distributed Data Handling: Non-relational databases excel in distributed environments, facilitating the storage and processing of data across multiple nodes or clusters. This distributed architecture enhances fault tolerance, resilience, and scalability, making non-relational databases well-suited for large-scale applications and distributed computing.

4. METHODS

In this section, we outline the methodologies employed in our comparative study of sentiment classification methods. We divide our approach into two main categories: machine learning methods and deep learning methods.

4.1. Machine Learning Methods

In this subsection, we delve into the realm of traditional machine learning algorithms applied to sentiment classification tasks. The advent of machine learning has paved the way for sophisticated techniques to discern sentiments from textual data, offering a diverse array of methodologies for analysis. Here, we explore and implement three prominent algorithms: K-Nearest Neighbors (KNN), Random Forest, and Multinomial Naive Bayes (MultinomialNB).

4.1.1. K-Nearest Neighbors

K-Nearest Neighbors (K-NN) is a supervised machine learning algorithm used for classification and regression tasks. These are the steps of the algorithm:
(i) Assign a value to K.
(ii) Calculate the distance between the new data entry and all other existing data entries. Arrange them in ascending order. Distances are normally used to measure the similarity or dissimilarity between two data objects

The Minkowski distance is a measure of the distance. By adjusting the 'p' value, the Minkowski distance transforms into well-known distance metrics. It

Fig. 1: KNN example.

is a generalization of other distance measures such as Euclidean distance and Manhattan distance.

$$d(P_1, P_2) = \left(\sum_{i=1}^{n} |x_{1i} - x_{2i}|^p \right)^{\frac{1}{p}} \quad (1)$$

where:

- $d(P_1, P_2)$ is the Minkowski distance between points P_1 and P_2,
- n is the dimension of the space (in this case, $n = 2$),
- p is a parameter that determines the degree of the Minkowski distance. When $p = 1$, it corresponds to the Manhattan distance; when $p = 2$, it corresponds to the Euclidean distance.

In the Minkowski distance formula, setting 'p' to 1 yields the Manhattan distance. The Manhattan distance measures the distance between two points by summing the absolute differences between their coordinates.

In the Minkowski distance formula, setting 'p' to 2 yields the Euclidean distance. The Euclidean distance calculates the shortest straight-line distance between two points in space.

(iii) Find the K nearest neighbors to the new entry based on the calculated distances.
(iv) Assign the new data entry to the majority class in the nearest neighbors.

How to choose the value of K?
Choosing the optimal value for K in K-nearest neighbor (KNN) is a critical challenge. A small K increases the risk of overfitting due to sensitivity to noise. Conversely, a large K can be computationally demanding and might dilute the concept behind KNN, which assumes that nearby points share similar classes. To find the best K, cross-validation can be used to test the KNN algorithm with various K values. The optimal K is the one that yields the highest accuracy in this cross-validation process.

Advantages of K-nearest neighbors algorithm:
- Knn is simple to implement.
- Knn executes quickly for small training data sets.
- Performance asymptotically approaches the performance of the Bayes Classifier.
- Don't need any prior knowledge about the structure of data in the training set.
- No retraining is required if the new training pattern is added to the existing training set.

Limitation to K-nearest neighbors algorithm:
- When the training set is large, it may take a lot of space.
- For every test data, the distance should be computed between test data and all the training data. Thus a lot of time may be needed for the testing.

4.1.2. Random Forest

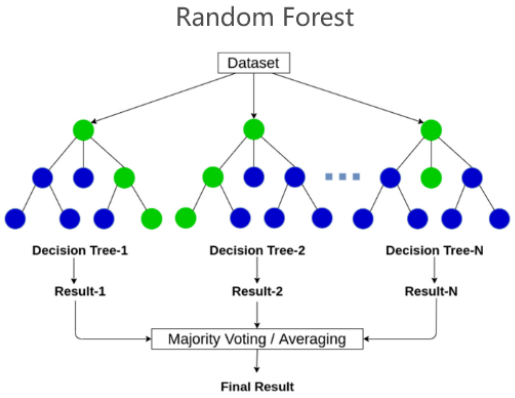

Fig. 2: Random Forest example.

The Random Forest algorithm is a widely used machine learning method for classification and regression. This section explores the operating principles of Random Forest as well as its applications and advantages.

(i) Initial training: Formation of a set of decision trees on bootstrap samples from the training dataset.
(ii) Construction of decision trees: Each tree is built by randomly selecting a subset of features at each decision node. The trees are grown until a stopping criterion is reached, such as maximum depth or a minimum number of observations in terminal nodes.
(iii) Prediction: Aggregation of predictions from each tree to obtain a final prediction. For classification, a majority vote can be used, while for regression, an average is often computed.
(iv) Feature importance: Measurement of the importance of each feature in prediction by evaluating the decrease in accuracy when the values of that feature are randomly permuted.

Random Forest offers several advantages that make it a widely used and versatile algorithm in machine learning. By combining multiple decision trees trained on different samples of data, Random Forest reduces the risk of overfitting compared to using a single decision tree, making it a robust choice for complex datasets. This ensemble approach also provides robustness to noisy data, as the aggregation of predictions helps mitigate the impact of outliers, ensuring more reliable outcomes even in imprecise environments.

Another key advantage is the ease of use; Random Forest does not require extensive hyperparameter tuning, unlike some other algorithms, making it accessible to both novices and experienced users. Scalability is also a strong suit, as Random Forest can efficiently handle large datasets with numerous features without overwhelming computational resources, maintaining reasonable execution times.

Additionally, Random Forest can handle missing data effectively, using techniques like imputation or variable exclusion during tree construction, reducing the need for elaborate preprocessing.

Thanks to these advantages, Random Forest has become one of the preferred algorithms for a range of classification and regression tasks, offering a balance of strong performance, robustness, and ease of use.

By combining these advantages, Random Forest has become one of the preferred algorithms for a variety of classification and regression tasks in the field of machine learning, offering both strong performance and ease of use.

4.1.3. Multinomial Naive Bayes

Multinomial Naive Bayes is a variant of the Naive Bayes algorithm that is specifically designed for discrete features,particularly for text classification tasks. It assumes that features follow a multinomial distribution, such as word counts in a document. The formula for Multinomial Naive Bayes can be represented as shown in Fig.3.

$$P(y|x_1, x_2, ..., x_n) = P(y) \times \prod_{i=1}^{n} P(x_i|y)^{x_i}$$

Where:

- $P(y|x_1, x_2, ..., x_n)$ is the probability of class y given the features $x_1, x_2, ..., x_n$.
- $P(y)$ is the prior probability of class y.
- $P(x_i|y)$ is the conditional probability of feature x_i given class y.

Fig. 3: Multinomial Naives Bayes formula

The use of Multinomial Naive Bayes has many benefits:

- **Efficiency**: Multinomial Naive Bayes is computationally efficient and can handle large datasets with many features which makes it a practical choice for text classification tasks like sentiment analysis.
- **Easy to implement**: Multinomial Naive Bayes is straightforward to implement and requires minimal hyperparameter tuning which makes it accessible for beginners and quick to deploy.
- **Robust**: The "naive" assumption of conditional independence can make Multinomial Naive Bayes robust to irrelevant features or noise in the data.

4.2. Deep Learning Methods

In this subsection, we explore the application of deep learning architectures to sentiment classification tasks. Deep learning has garnered significant attention in recent years for its ability to automatically learn intricate representations from raw data, making it particularly well-suited for tasks involving complex patterns and dependencies, such as sentiment analysis.

Deep learning methodologies, inspired by the structure and function of the human brain's neural networks, have revolutionized the field of artificial intelligence. These techniques excel at learning hierarchical representations of data, capturing intricate relationships and patterns that may be challenging to discern using traditional machine learning algorithms. In the realm of sentiment analysis, deep learning models offer the promise of improved performance and flexibility, enabling the automatic extraction of features from raw text data without the need for manual feature engineering.

4.2.1. CNN

Convolutional Neural Networks (CNNs) are the most widely used deep learning architectures in image processing and image recognition. It also works particularly well when dealing with data sequences, like text. CNNs can be used in NLP to perform text classification by applying several filters to the data to find patterns and features in the input data and then using those patterns and features to generate predictions or decisions.

4.3. TextBlob Library

The sentiment analysis of Bitcoin-related tweets was carried out using a pretrained Naive Bayes classifier from the TextBlob library Loria (2023). This classifier assigns a polarity score to each tweet, which ranges from -1 (indicating a negative sentiment) to 1 (indicating a positive sentiment), with 0 as the midpoint for neutral sentiment. These polarity scores were incorporated into our dataset. Although the classifier also provides a subjectivity score, indicating how subjective the input text is, we did not utilize this output in our analysis.

5. IMPLEMENTATION

In this section, we describe how we implemented various machine learning and deep learning methods for sentiment classification. We will explain the steps followed to prepare the data, train the models, and evaluate their performance.

We'll start by detailing traditional machine learning techniques like K-Nearest Neighbors (KNN), Random Forest, and Multinomial Naive Bayes.

Next, we'll discuss deep learning methods such as Convolutional Neural Networks (CNNs).

5.1. Machine Learning

The objective of this section is to describe the implementation of machine learning methods for sentiment classification. We focus on three main algorithms: K-Nearest Neighbors (KNN), Multinomial Naive Bayes, and Random Forest. We will explain how these algorithms were configured, trained, and tested with a dataset, emphasizing feature extraction, data splitting, and performance evaluation.

5.1.1. Data Preparation

To start, the reviews and their associated labels were extracted as follows: reviews contains the list of reviews. labels contains the corresponding labels (positive, negative or neutral).

5.1.2. Text Vectorization

We used CountVectorizer to transform the text into numerical representations, called features. This involves converting raw text into a matrix where each column represents a unique word (tokens), and the values indicate the frequency of that word in each review. This is a crucial step to make the data usable by machine learning algorithms.

5.1.3. Data Splitting into Training and Test Sets

To verify the performance of the algorithms, we split the vectorized data into training and test sets. The split was as follows:

- 80% of the data was used for training $(X_{\text{train}}, y_{\text{train}})$.
- 20% of the data was used for testing $(X_{\text{test}}, y_{\text{test}})$.

This split was carried out using `train_test_split` with a `random_state` set to 42 to ensure reproducibility.

5.1.4. K-Nearest Neighbors

For KNN, we used the parameter $n\_neighbors = 10$, which means the algorithm considers the 10 closest neighbors to determine the class of a given review.

Training: The KNN algorithm was trained with `fit(`X_train, y_train`)`.

Prediction: Predictions on the test set were made with `predict(`X_test`)`.

Evaluation: Accuracy was calculated with `accuracy_score(`$y\_test, knn\_predictions$`)`.

5.1.5. Random Forest

Training: The Random Forest model was trained with fit(X_train, y_train).

Prediction: Predictions on the test set were made with `predict(`X_test`)`.

Evaluation: Accuracy was calculated with `accuracy_score(`$y\_test, rf\_predictions$`)`.

5.1.6. Multinomial Naive Bayes

Training: The Naive Bayes model was trained with `fit(`X_train, y_train`)`.

Prediction: Predictions on the test set were made with `predict(`X_test`)`.

Evaluation: Accuracy was calculated with `accuracy_score(`$y\_test, nb\_predictions$`)`.

5.2. Deep Learning

In this section we will focus on the deep learning approach. Specifically, we will implement the dynamic CNN approach from Erkan and Gungor. As this paper stated, several processing and tokenisation combination exist and yield different result. We decided to work on a integer encoding with a word - preprocessed tokenization.

5.2.1. Preprocessing

The first step consist of removing none desired characters such as escape character, HTML tags, hashtags or empty string. We performed this step as follow :

- Lower case the content
- Removing empty comment.
- Removing special characters such as HTML tags, hashtags, '@',...
- Removing other non-alphanumeric characters.
- Removing punctuation marks except for '?'

5.2.2. Tokenization

The tokenization process involves breaking down text into individual tokens to uncover insights and extract language features. While various approaches exist, our focus was on employing a straightforward word tokenization method using the NLTK library.

5.2.3. Embedding

In this step, we try to find a representation of text as vectors. We use simple integer word embedding where each word is assigned to a integer. Same word have the same integer value. We also limit the size of the vectors to the average length of the comment (ie. 234 for IMBD and 148 for AMAZON). We than proceed to add padding to our vector by adding zeros so each vector has the same length.

5.2.4. Data splitting

We partitioned our dataset into three subsets: 80% for training, 10% for validation, and 10% for testing purposes. Notably, our dataset exhibited a balanced distribution across classes, ensuring that each class was represented fairly. This balance is crucial as imbalanced class distributions could potentially introduce bias into our analyses.

5.2.5. Label Encoding

To encode sentiment classes into vector representations, we leveraged the Keras library. For binary sentiment classification (positive and negative), we utilized vectors of length 2. In scenarios involving sentiment classification with three distinct classes, we employed vectors of length 3 to represent the classes.

5.2.6. Model architecture

The model follow a similar architecture than the one proposed by Erkan and Gungor (Fig.4), but differ based on the dataset (binary vs multi-classes classification).

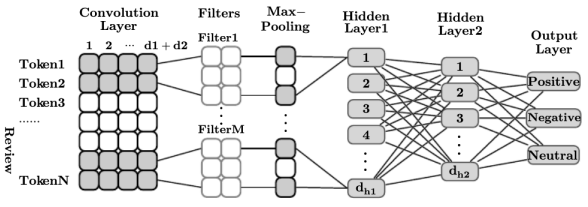

Fig. 4: CNN model with Word2Vec and Glove embeddings.

For the IMDB dataset, we initiate our model with an embedding layer. This layer facilitates the transformation of sparse and high-dimensional word rep-

resentations into compact, lower-dimensional embeddings.

The subsequent layer in our architecture is a 1D convolutional layer of size 50. This layer employs Rectified Linear Unit (ReLU) activation, allowing the network to introduce non-linearity and effectively learn complex patterns from the embedded representations of the input data.

Following the convolutional layer, we introduce a MaxPooling layer. This layer serves to down-sample the feature maps obtained from the convolutional layer, retaining only the most salient information while reducing computational complexity. The MaxPooling layer is succeeded by a hidden layer comprising 8 units, providing further abstraction and representation learning capacity to the model.

In terms of optimization, we employ the Adam optimize, which is well-suited for training deep learning models on text data.The output layer use the simgoid activation function. Additionally, we utilize binary cross-entropy as the loss function, which is a standard choice for binary classification tasks such as sentiment analysis.

For the Amazon review dataset, we use the same architecture but modify some parameters. The first Convolution Layer is of size 300, we use categorical crossentropy as the loss function and softmax activation function for the output layer.

5.3. Bitcoin Sentiment Analysis

The following section outlines our implementation of sentiment analysis on Bitcoin-related tweet data, referencing the dataset described in Section 3.3. We detail the preprocessing steps undertaken and the methodology for storing the data in a data management solution. This preparation facilitates subsequent sentiment analysis conducted with the TextBlob library Loria (2023). The culmination of this analysis is the construction of a Bitcoin sentiment-price graph, which illustrates the correlation between the analyzed sentiment and the Bitcoin market price at corresponding times.

Accompanying this analysis is a Jupyter Notebook Kaibel (2023), which provides a comprehensive code-based walkthrough of the entire process.

5.3.1. Data Reduction, Preprocessing and Management

Due to resource constraints, we implemented several data reduction strategies to adhere to our hardware specifications. This involved eliminating all columns from the dataset except for "text," "hashtags," and "date." Additionally, we filtered out tweets containing hashtags that suggested promotional offers or advertisements, as these could have skewed the data for subsequent analysis. After this filtering, the hashtags column was no longer necessary and was therefore dropped. To further meet our database size constraint of 250MB, we utilized only 15% of the available dataset.

In terms of preprocessing, we removed special characters from the text of each tweet to prevent potential confusion in the later stages of sentiment analysis modeling. For storing the preprocessed data, we chose MongoDB (2024) due to its intuitive interface and seamless integration with our Python environment. The data upload to MongoDB (2024) was executed in chunks to avoid timeouts that occurred with larger simultaneous uploads, ensuring efficient and reliable data storage.

5.3.2. Sentiment Analysis

For the sentiment analysis, we utilized the preprocessed data stored in MongoDB (2024) from our previous efforts to extract the polarity score of each tweet, as detailed in Section 4.3. We then gathered historical Bitcoin market prices for the time frame of the tweet data using the Yahoo Finance API Yahoo Finance (2023). To synthesize this information, we calculated the average of all daily polarity scores, creating a dataset that contains a single entry for each date, which pairs the average polarity score with the corresponding Bitcoin market price. This dataset was then visualized in a graph, as shown in Figure 5.

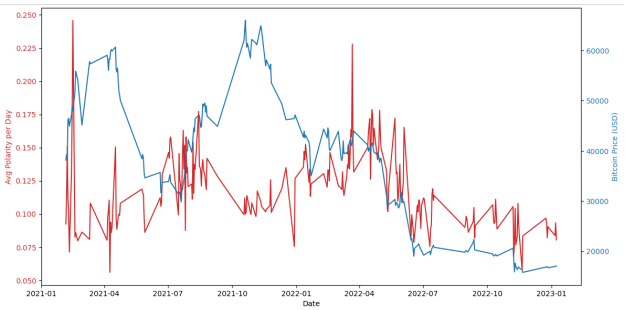

Fig. 5: Bitcoin Sentiment and Price from 02/2021 to 02/2023

6. EXPERIMENTAL RESULT AND ANALYSIS

In this section, we present the results obtained from machine learning and deep learning algorithms tested on two different datasets: Amazon and IMDB. Additionally, we include the outcomes obtained using TextBlob

6.1. Machine Learning Result

This section presents the experimental results obtained with various machine learning algorithms on

two datasets: Amazon and IMDB. The performance of the algorithms is measured in terms of accuracy. We used three main algorithms: K-Nearest Neighbors (KNN), Multinomial Naive Bayes, and Random Forest.

6.1.1. Amazon Dataset

The Amazon dataset contains product reviews from the Amazon platform. The results show that the Naive Bayes algorithm achieved the highest accuracy on this dataset, reaching about 67.2%. Random Forest came in second with an accuracy of 63.78%. KNN had the lowest accuracy, at about 49.9%.

6.1.2. IMDB Dataset

The IMDB dataset contains movie reviews. In the case of the IMDB dataset, Random Forest achieved the best accuracy with 85.71%. Naive Bayes obtained an accuracy of 84.87%, and KNN achieved an accuracy of 66.02%.

6.1.3. Confusion Matrix

The results of the machine learning methods are showed in a confusion matrix for IMDB and Amazon datasets. The confusion matrix is a matrix that measures the quality of a classification system. Each row corresponds to a real class, each column corresponds to an estimated class.

For the IMDB confusion matrix (Fig.6), the KNN method gives worse results than Naives Bayes and Random Forest. The true positive and true negative values are better for Naives Bayes and Random Forest. 87,89% true positive for Naives Bayes and 85,73% for Random Forest while KNN has only 65,47%. The true negative are approximately the same. This means Naives Bayes and Random Forest can correctly classify a review as positive or negative while KNN struggles a bit.

For the Amazon matrix confusion (Fig.7), the KNN method struggles to classify positives reviews (32,17%) and negatives reviews (46,04%) while it manages to classify with a good rate the neutral one (71,21%). For Naives Bayes and Random Forest confusion matrix the scores are approximately the same, like for previous dataset. Naives Bayes can classify a positive review in the right class with 66,91% and Random Forest with 63,94%. For the negatifs ones, the score are the same 59,66% for Naives Bayes and 62,52% for Random Forest. Naives Bayes has the best rate for neutrals reviews with 75,25%. We can conclude that when there are more classes, theses methods have a little more difficulty to classify datas. Naives Bayes and Random Forest have overall better classification rates.

6.1.4. Accuracy Evolution With Data Size

In this section, we analyze how the accuracy of different algorithms evolves with increasing dataset size. The associated graph (Fig.8) shows the accuracy trend for each algorithm, both on the training set and the test set.

K-Nearest Neighbors shows an upward trend, but its growth is slower than the other algorithms. This might be because KNN is more sensitive to data density and the distance between data points. Although KNN improves with more data, it is likely to suffer from limitations due to increasing complexity as the dataset grows.

Random Forest shows a continuous increase in accuracy as the dataset size grows. This trend is consistent with the robust nature of the algorithm, which typically improves with more training data. Although Multinomial Naive Bayes (MNB) initially has a higher accuracy, the graph suggests that Random Forest will surpass MNB at some point, indicating that with enough data, RF becomes the most performant model.

Multinomial Naive Bayes has a relatively strong start in terms of accuracy. This might be due to its efficiency on smaller datasets and its simplicity. However, its growth curve is more stable compared to Random Forest, suggesting that this algorithm might reach a performance ceiling with moderately sized datasets, with improvement slowing beyond a certain point.

6.2. Deep Learning Result

In this section we will focus on the result obtained by CNN on the two datasets.

6.2.1. Amazon Dataset

The Amazon dataset yielded an accuracy of 74%. Analysis of the confusion matrix (see Fig. 9) reveals that the neutral label contributes significantly to misclassifications in the prediction of labels. The complexity to classify neutral sentiment is not surprising and can be improved by further data annotation of this class and training.

6.2.2. IMBD Dataset

The CNN architecture yielded promising results with an accuracy of 88%. While this accuracy is slightly lower than the top performance of 93% from Erkan and Gungor, it's important to note that our model is significantly simpler. This trade-off between model complexity and performance highlights the balance we've struck in designing a model that is both effective and efficient for sentiment analysis on the IMBD dataset. The confusion matrix for the

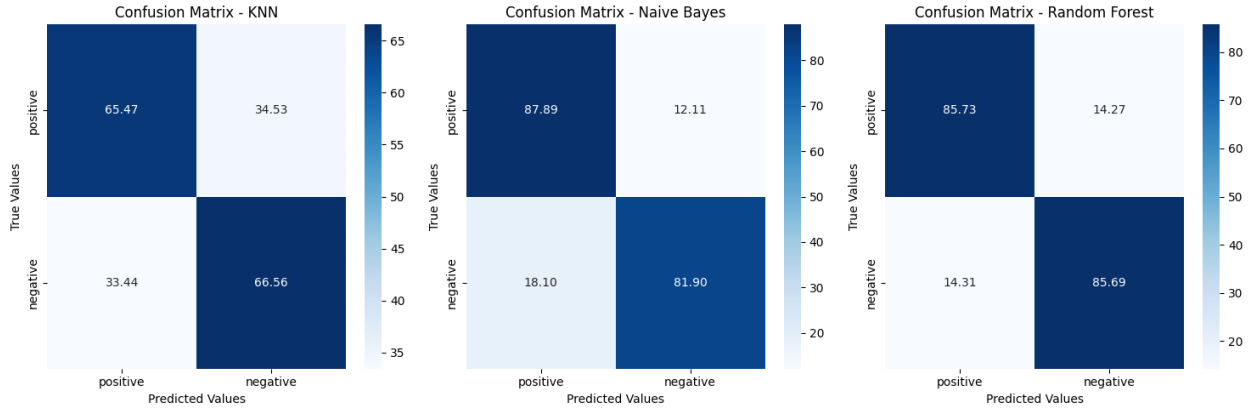

Fig. 6: Confusion Matrix ML IMBD

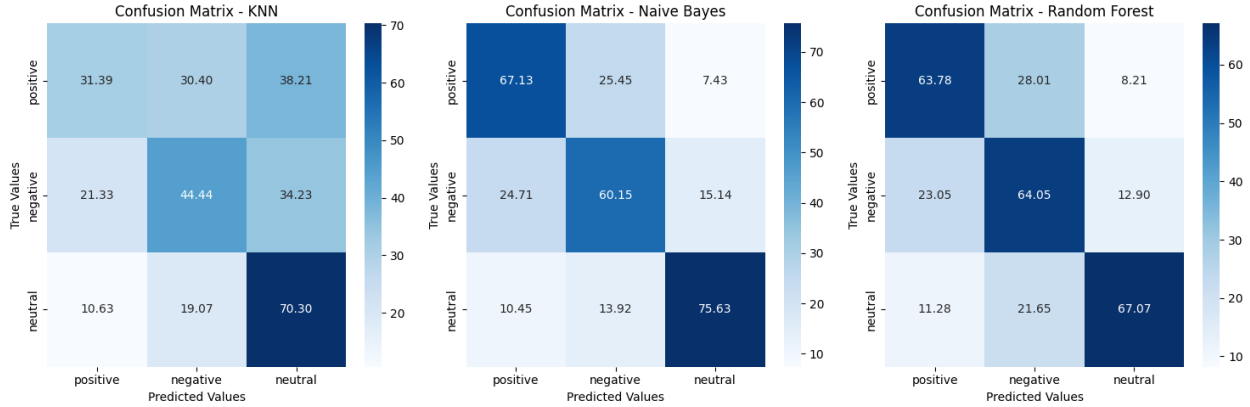

Fig. 7: Confusion Matrix ML Amazon

IMBD dataset (Fig.10) highlight the overall good performance of the model.

6.3. Comparison

The following table summarizes the accuracy results. Although CNN achieves the highest accuracy in both cases, Random Forest also performs very well, particularly on the IMDB dataset. The detailed results are provided in Table 1.

Table 1: Accuracy Result

Model	Accuracy	Dataset
KNN	0.499	Amazon
Random Forest	0.6378	Amazon
MultinomialNB	0.672	Amazon
CNN	**0.74**	Amazon
KNN	0.66	IMDB
Random Forest	0.8571	IMDB
MultinomialNB	0.8487	IMDB
CNN	**0.88**	IMDB

6.4. Bitcoin Sentiment Results

Our results in Fig. 5 indicate that public sentiment towards Bitcoin, particularly within the Twitter community, generally leans positive, as the polarity scores do not reach neutral or negative levels. Nonetheless, a correlation is observable between changes in public sentiment and Bitcoin prices, except during early and late 2021 when Bitcoin's price reached its highest values within the analyzed timeframe. This suggests that while sentiment usually shifts in correlation with Bitcoin's price movements, significant price increases often provoke more skeptical tweets.

Interestingly, as these price spikes began to recede, the sentiment became more positive, suggesting that even as market prices declined, optimism about Bitcoin's future persisted. Perhaps this enduring belief in the technology behind Bitcoin, even during its challenging phases, may be what sustains its lasting significance in the cryptocurrency world.

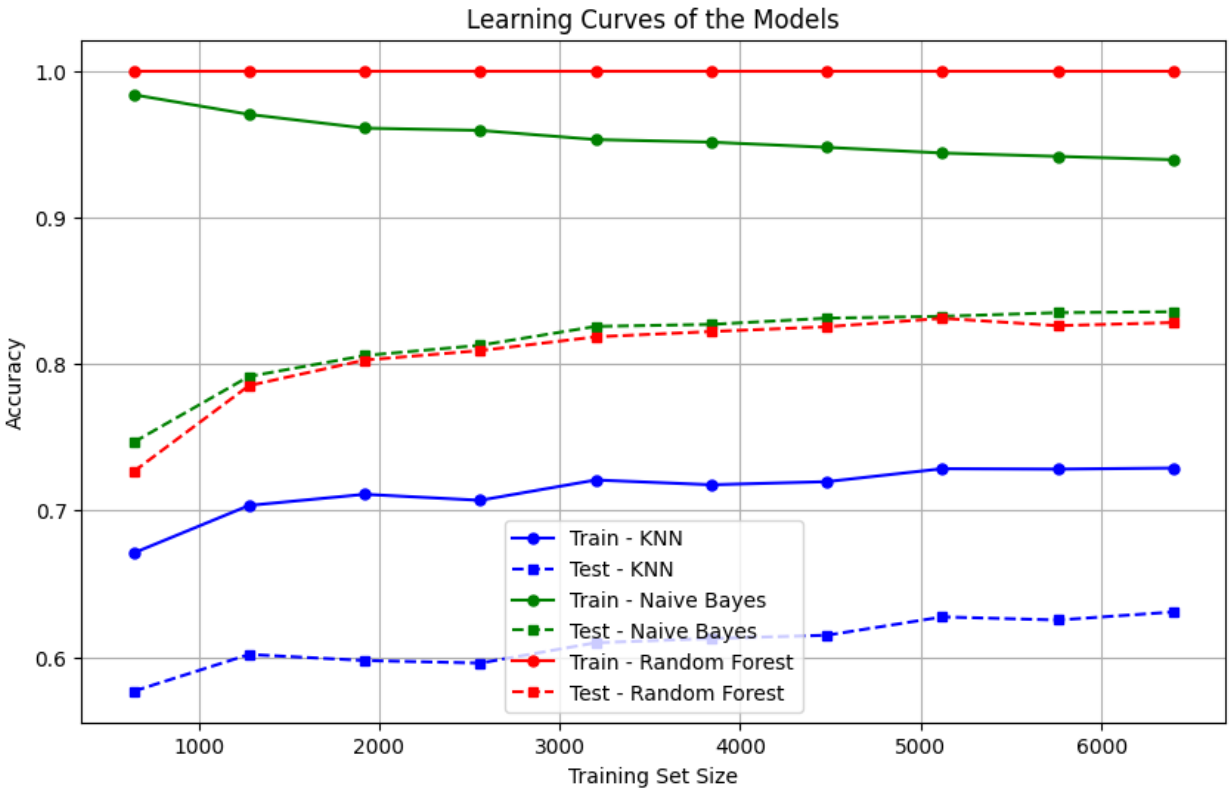

Fig. 8: Accuracy Evolution

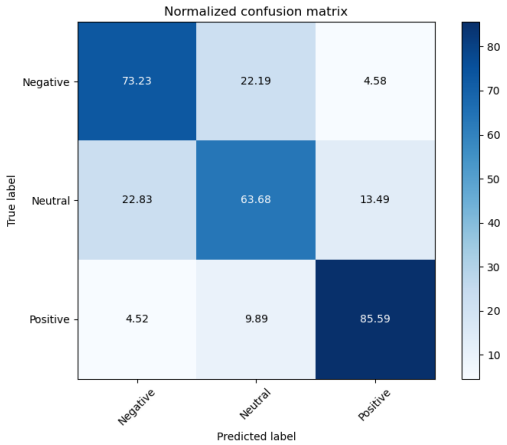

Fig. 9: Amazon confusion matrix

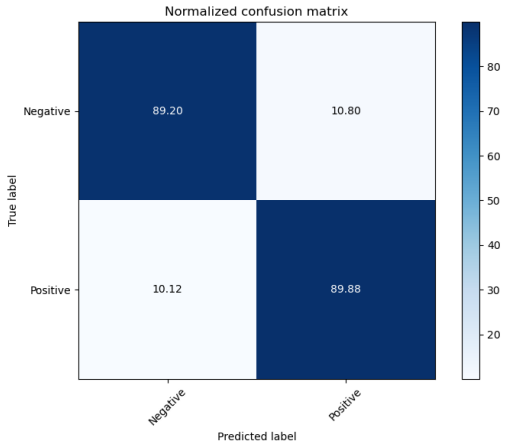

Fig. 10: IMBD confusion matrix

7. CONCLUSION

In summary, deep learning has demonstrated superior performance compared to traditional machine learning methods in sentiment classification tasks. Deep learning architectures, such as convolutional neural networks (CNN), are particularly effective at identifying complex patterns and dependencies in textual data, leading to more accurate results in sentiment analysis.

However, the Random Forest method has also proven to be a robust and reliable alternative to deep learning, offering good accuracy with a much faster training process. This can be especially valuable in scenarios where computational resources are limited or when rapid results are needed. The simplicity and

speed of Random Forest make it an excellent choice for applications where the computational demands of deep learning might be impractical.

An interesting approach to sentiment classification involves using libraries that offer more advanced techniques or pre-trained models. These tools can significantly reduce the time and effort required to build effective sentiment analysis systems. By leveraging pre-trained models or specialized libraries, practitioners can achieve robust results without the high computational costs typically associated with training deep learning models from scratch.

Thus, while deep learning is ideal for achieving the highest possible accuracy, Random Forest can be preferable in some cases due to its speed and lower computational requirements. Practitioners should consider the specific needs of their use case, the availability of resources, and the desired training speed when choosing between these approaches for sentiment classification. Additionally, utilizing libraries with advanced methods or pre-trained models can be a strategic choice, providing a balance between performance and resource efficiency.

REFERENCES

Durga, P. and Godavarthi, D. 2023, IEEE Access, PP, 1

Erkan, A. and Gungor, T. 2023, IEEE Access, PP, 1

Feizian, F. and Amiri, B. 2023, IEEE Access, 11, 142177

Hu, A. and Flaxman, S. R. 2018, in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018, ed. Y. Guo and F. Farooq (ACM), 350–358

Iosifidis, V. and Ntoutsi, E. 2017, in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017 (ACM), 1823–1832

Kaibel, L. 2023, Bitcoin Sentiment Analysis Jupyter Notebook, accessed: 2024-04-23

Loria, S. 2023, TextBlob: Simplified Text Processing, python library for processing textual data

MongoDB. 2024, MongoDB, accessed: 2024-04-24

Perti Ashwin, S. A. and Ankit, V. 2023, , 2375

Suresh, K. 2023, Bitcoin Tweets Dataset, accessed: 2024-04-23

Yahoo Finance. 2023, Yahoo Finance API, access to financial data API