Group Number: **07**

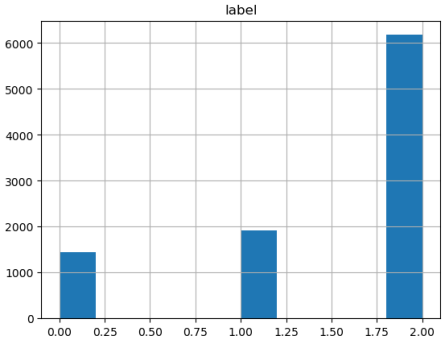
Group Members: **Francisco Perestrello, 20241560@novaims.unl.pt**

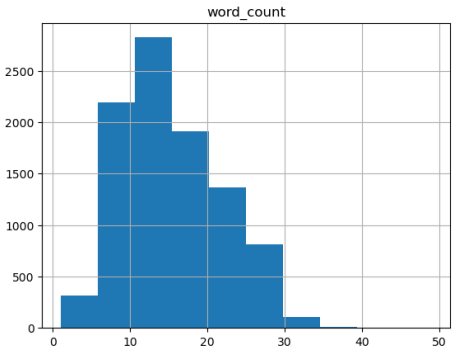
1. Introduction

The following report aims to develop a natural language processing (NLP) model to predict market sentiment based on Twitter data. Each tweet will be labeled with one of three sentiment categories: Bearish (0), Bullish (1), or Neutral (2). The study will explore various embedding methods to represent textual data, starting with simpler techniques and progressing to more sophisticated approaches. In parallel, several classification models will be employed to analyze their predictive capabilities, ranging from traditional algorithms to more advanced architectures. This comprehensive approach will provide insights into the effectiveness of different methods for sentiment prediction in financial contexts, in a quest to find the model that best predicts sentiment.

2. Data Exploration

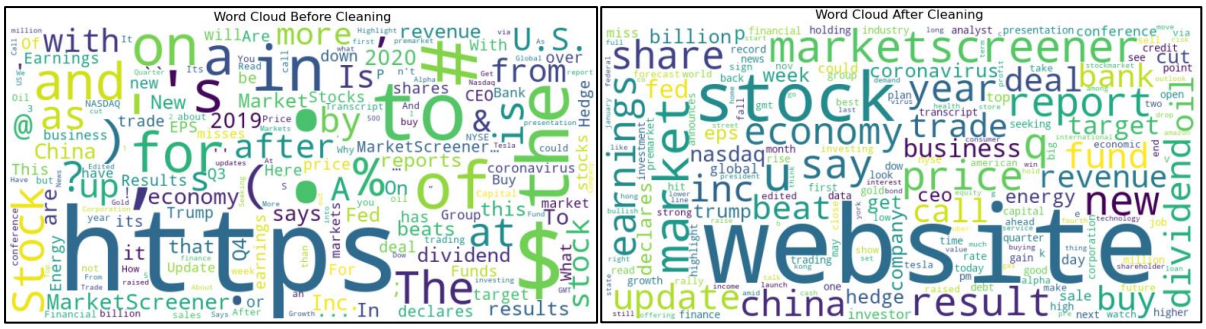
The training dataset consists of 9,543 observations, with no missing values ensuring a complete analysis. The dependent variable, representing the sentiment labels, can take one of three values: 0 for Bearish, 1 for Bullish, or 2 for Neutral. The histogram of the distribution reveals that the data is imbalanced, with the majority of tweets classified as Neutral, outnumbering the combined total of Bearish and Bullish tweets.





Using nltk's word_tokenize, we counted the number of words in each tweet before any preprocessing was applied. The independent feature ranges from a minimum of 1 word to a maximum of 49, with the majority of tweets containing between 10 and 20 tokens. This provides an initial understanding of the textual content's variability.

Calculating token frequency, the resultant word cloud highlights that website symbols (e.g., "http", "$", "#") and common linguistic connectors (e.g., "to", "of", "the") dominate the space, suggesting the need for cleaning to better capture meaningful patterns. After applying a cleaning process, which will be detailed in the following sections, the word cloud becomes more interpretable. Key terms related to the financial market, such as "stock," "market," "dividend," and references to websites, emerge as dominant themes, providing valuable context for sentiment analysis.



3. Data Preprocessing

The corpora were divided into training and validation sets, with the validation set comprising 10% of the training data to ensure a robust evaluation of the model's performance.

Several preprocessing methods were applied to clean the data. Initially, all text was converted to lowercase to ensure consistency. Next, the most common regular expressions, which are patterns used to identify specific text sequences, were employed to replace website-related content with the placeholder word "website". Numerical data and punctuation were subsequently removed to eliminate unnecessary noise. Finally, stopwords - commonly used words in the English language that typically carry little meaning in the context of sentiment analysis - were also removed.

Extra Work

Two additional preprocessing methods were developed: lemmatization and stemming. Lemmatization involves reducing words to their base or root form while retaining their semantic meaning. In contrast, stemming focuses on trimming words to their root by removing affixes, sometimes sacrificing readability or semantic clarity. For this project, lemmatization was chosen to reduce noise while maintaining semantic coherence, which is crucial for downstream tasks involving advanced embeddings and pre-trained sentiment classification models.

4. Feature Engineering

In the feature engineering stage, various embedding methods were applied to transform the cleaned text data into numerical representations suitable for modeling. For all methods, models were trained exclusively on the training set and subsequently applied to the validation set to prevent data leakage.

The first approach employed was the Bag of Words (BoW). This method involves creating a vocabulary from the training corpus and representing each document as a vector indicating the occurrence of each word in the vocabulary. The resulting vectors are shaped by the number of observations and the number of unique words in the training corpus, leading to a high-dimensional array. While simple and effective for some tasks, this method does not account for semantic relationships between words.

The second approach applied was Term Frequency-Inverse Document Frequency (TF-IDF), which extends BoW by weighting words based on their importance. Words frequently appearing in a document but rarely across the entire corpus receive higher weights, reflecting their relevance. For this project, both unigrams (single words) and bigrams (two-word sequences) were extracted from the dataset, leading to an array with dimensions five times larger than that of BoW. This highlights a major challenge of these

simpler methods: they can quickly expand into extremely high-dimensional representations. Additionally, both BoW and TF-IDF fail to capture semantic relationships and the contextual importance of words.

To address these limitations, a word embedding approach was implemented using Word2Vec. Unlike BoW and TF-IDF, Word2Vec represents each word as a dense vector in a continuous vector space, capturing semantic and contextual relationships between words. For this project, pre-trained GloVe (Global Vectors for Word Representation) embeddings - specifically the glove-twitter-25 vectors - were utilized. Each document was transformed into an array composed of the number of tokens and the size of the embeddings, 25, resulting in three-dimensional dense arrays for the entire training and validation sets.

While Word2Vec effectively transforms textual data into dense vector representations, it introduces a challenge for model compatibility. Since each document has its own number of tokens, the resulting dense array has inconsistent dimensions, making it unsuitable for models to fit directly. To address this, pad sequencing was applied. Pad sequencing ensures that all documents have the same number of tokens by padding shorter sequences with zeros and truncating longer ones, and it was applied using the maximum token length from the training data as a threshold. This process converts the variable-length sequences into fixed-length ones, allowing them to be structured as three-dimensional tensors. However, even after achieving consistency, this three-dimensional dense array cannot be processed by simpler machine learning models, which typically require two-dimensional input. To resolve this, a two-dimensional array was created by averaging the word embeddings for each document. This transformation resulted in an array where each document is represented by a single vector of size equal to the embedding dimensions, ensuring compatibility with models like Logistic Regression and KNN at the cost of some lost semantic information.

Extra Work

In an attempt to enhance the representation of the textual data, two additional feature engineering methods were employed. The first approach utilized a Transformer model, specifically the pre-trained embedding language model 'Twitter XLM Roberta Base Sentiment', designed to analyze tweet sentiment. Transformer models are a class of deep learning models that rely on self-attention mechanisms to process sequences of text. They excel in capturing contextual relationships between words across long distances, making them highly effective for natural language processing tasks. Using this model, each document in the corpora was represented as an array comprising the number of tokens in the text and the size of the embeddings (set to 768 in the language model), resulting in our training and validation sets becoming three-dimensional dense arrays. This approach preserves both the sequence and context of the text, offering a rich and nuanced representation for sentiment classification. Much like Word2Vec, these vectors are inconsistent in size and require the same two-step preprocessing of pad sequencing followed by averaging the word embeddings for each document.

The second method applied was Point-wise Mutual Information. PMI measures the strength of association between two words by comparing their co-occurrence probability to the probabilities of their independent occurrences. This approach identifies meaningful word associations, such as common bigrams, by calculating PMI scores for all bigrams in the corpus. Using the nltk package, training bigrams were extracted, their scores were computed, and a feature vector was created for each document based on these scores. Finally, these vectors were standardized using a standard scalar from sklearn. While PMI can uncover valuable word associations, the resultant arrays suffer from the same dimensionality issues as Bag of Words and TF-IDF, producing large, sparse arrays that can challenge computational efficiency and model performance.

5. Classification Models

Several classification models were tested to evaluate their performance in predicting market sentiment through tweets. The first model run was K-Nearest Neighbors. KNN is a simple yet effective classification algorithm that assigns a label to a data point based on the majority class of its nearest neighbors in the feature space. For this project, 10 neighbors were used, and the cosine distance metric was applied. Cosine distance measures the angular difference between vectors, making it particularly suitable for text data since it emphasizes the directionality of embeddings rather than their magnitude, which is crucial in high-dimensional sparse datasets.

The second model tested was Logistic Regression. This model uses a linear combination of input features and applies a logistic function to predict probabilities for each class. Logistic Regression is robust for binary and multi-class classification problems, and its simplicity allows for quick training and effective baseline performance evaluation. Notably, some more advanced embedding methods might introduce multicollinearity problems.

Next, a Naive Bayes model was applied. Naive Bayes is a probabilistic classifier based on Bayes' Theorem, which assumes conditional independence between features given the class label. Despite its simplicity and the independence assumption, Naive Bayes often performs well with text data, as word frequencies tend to satisfy this assumption reasonably well in practice. Nonetheless, embedding methods that violate this assumption are expected to perform worse across the validation dataset.

Finally, a Multi-Layer Perceptron was tested. An MLP is a feed-forward neural network model that captures complex non-linear relationships in the data. For this project, the MLP was configured using two hidden layers, each containing two neurons, and utilized the 'adam' solver, which is an adaptive gradient-based optimizer known for its efficiency in training deep learning models. The ReLU (Rectified Linear Unit) activation function was applied to introduce non-linearity, which helps the network learn more complex patterns. These settings were chosen to balance computational efficiency and model expressiveness while avoiding overfitting.

For the more advanced embedding methods, such as Word2Vec and Transformer-based embeddings, the two-dimensional versions of the arrays were used to fit these models, ensuring compatibility across all approaches. This allowed for a consistent comparison of classification performance across different feature engineering methods, even if it results in reduced performance.

Extra work

Two additional classification models were applied to further explore the performance of advanced methodologies.

The first model tested was a Pre-Trained Large Language Model (LLM), specifically the twitter-xlm-roberta-base-sentiment. Large Language Models, like transformers, are designed to process text data in its raw form using their pre-trained weights, capturing complex contextual and semantic information. This model was applied using its text classification capabilities, directly predicting sentiment labels based on the cleaned text data. Unlike other models that require pre-computed embeddings, the LLM bypasses the need for manual feature engineering by leveraging its pre-trained knowledge, streamlining the workflow for advanced NLP tasks.

The second model applied was a Long Short-Term Memory Neural Network (LSTM). LSTMs are a type of recurrent neural network specifically designed to handle sequential data, making them highly suitable for text-based applications. They excel in capturing long-term dependencies and temporal patterns in sequential inputs, such as word embeddings or tokenized sequences. The LSTM model was applied to the dense and more complex inputs derived from Word2Vec and the Transformer embeddings, as these representations preserve semantic relationships and contextual dependencies across tokens, which LSTMs can effectively process. For the Word2Vec embeddings, the input size was set to match the embedding dimension of 25, with the model structured using 6 layers and totaling 5,475 trainable parameters. For the Transformer embeddings, which have an input size of 768, the same architecture with 6 layers was used, resulting in a model with 100,579 trainable parameters. In both cases, the 'softmax' activation function was chosen to process the multi-class problem with the ‘adam’ optimizer.

By incorporating these sophisticated classification models, the project evaluates a broad spectrum of approaches, ranging from traditional machine learning to cutting-edge deep learning methodologies, ensuring a comprehensive analysis of sentiment prediction strategies.

6. Evaluation and Results

K-Nearest Neighbors (KNN)

A comparative analysis of the classification reports of the different embedding methods reveals that the TF-IDF approach achieved the best performance when paired with the KNN model, yielding a weighted average f1 score of 0.77. The f1 score is a harmonic mean of precision and recall, providing a balanced measure that accounts for both false positives and false negatives. This makes it particularly useful for evaluating imbalanced datasets like ours.

TFIDF

	precision	recall	f1-score	support
Bearish	0.68	0.39	0.50	140
Bullish	0.63	0.58	0.60	176
Neutral	0.83	0.92	0.87	639
accuracy			0.78	955
macro avg	0.71	0.63	0.66	955
weighted avg	0.77	0.78	0.77	955

While most models demonstrated high recall scores for neutral sentiment - a measure of how effectively a model identifies all actual instances of a particular class - TF-IDF excelled in capturing the less frequent bearish and bullish sentiments. With a weighted average recall score of 0.78, it outperformed other methods in addressing the class imbalance challenge. Moreover, the model achieved the highest weighted average precision, which measures the proportion of correctly predicted observations per label, and returned the best accuracy with a score of 0.78, representing the percentage of correctly classified instances out of the total observations.

Interestingly, the Bag of Words approach yielded results very close to those of TF-IDF, further underscoring the compatibility of these simpler embedding methods with the KNN model. Their ability to produce meaningful feature spaces for a straightforward classification model like KNN highlights their suitability in this context.

Logistic Regression

For the Logistic Regression model, the Bag of Words approach stood out, achieving a weighted average f1 score of 0.79, the highest among all embeddings tested. It also delivered the best performance across precision, recall, and accuracy, in weighted average terms. This superior performance can be attributed to Logistic Regression's strength in handling sparse, high-dimensional data like BoW, where each word is treated as an independent feature. The linear nature of Logistic Regression aligns well with the feature representation provided by BoW, allowing it to effectively separate classes based on word frequencies.

BoW

	precision	recall	f1-score	support
Bearish	0.74	0.53	0.62	140
Bullish	0.64	0.64	0.64	176
Neutral	0.85	0.90	0.87	639
accuracy			0.80	955
macro avg	0.74	0.69	0.71	955
weighted avg	0.79	0.80	0.79	955

As observed with the KNN model, TF-IDF and BoW showed very similar results under Logistic Regression. Both methods rely on frequency-based representations, which Logistic Regression is well-equipped to handle, further demonstrating their suitability for simpler, linear classification algorithms.

In contrast, the Word2Vec embeddings performed poorly with Logistic Regression. This can be attributed to Word2Vec's dense and continuous feature representations, which may not align well with the linear decision boundaries of Logistic Regression, and introduce the problem of multicollinearity. The complexity and interdependencies captured by Word2Vec are better suited for non-linear models like neural networks, which can exploit the deeper structure in the embeddings. Notably, the PMI approach yielded the highest neutral sentiment recall (0.97) of all combinations, i.e., of all actually neutral observations, the model was able to capture 97% of them.

Naive Bayes

Under the Naive Bayes classification model, the TF-IDF embedding method finally diverged from the Bag of Words approach and emerged as the clear winner, showing a 0.71 f1 score. It achieved a weighted average precision of 0.72, recall of 0.71, and accuracy of 0.71, outperforming all other embedding methods. TF-IDF's ability to weigh terms based on their importance in the document relative to the entire corpus aligns well with Naive Bayes' probabilistic framework, as it helps emphasize distinguishing words for sentiment classification.

TFIDF

	precision	recall	f1-score	support
Bearish	0.45	0.52	0.48	140
Bullish	0.53	0.53	0.53	176
Neutral	0.83	0.80	0.81	639
accuracy			0.71	955
macro avg	0.60	0.62	0.61	955
weighted avg	0.72	0.71	0.71	955

In contrast, the Word2Vec embeddings struggled to deliver competitive results. This poor performance can be attributed to two main factors. First, Word2Vec violates the independence assumption inherent to Naive Bayes, as it generates dense, continuous vectors where the dimensions are interdependent and capture semantic relationships between words. Naive Bayes, however, assumes that features are conditionally independent given the class, a key assumption that Word2Vec violates. Secondly, Word2Vec embeddings are designed to encode semantic similarities, making them better suited for models that can capture non-linear relationships and context, such as neural networks. Naive Bayes, being a simple probabilistic classifier, is not equipped to leverage the richer, contextual information encapsulated in Word2Vec embeddings, further limiting its effectiveness in this scenario.

Multi-Layer Perceptron

The results from the Multi-Layer Perceptron model were intriguing, as three embedding methods - Bag of Words, TF-IDF, and the pre-trained embedding model - achieved the highest weighted average f1 score of 0.71. Among these, TF-IDF stood out for its precision, while the transformer-based pre-trained embedding model edged ahead in both recall and accuracy. This highlights the ability of different embeddings to leverage distinct strengths within the MLP framework, with TF-IDF excelling in precision-oriented tasks and the pre-trained model capitalizing on context-rich representations. However, the MLP struggled with Word2Vec embeddings, failing to converge and resulting in suboptimal performance. This issue likely stems from the small embedding size (25 dimensions) used with Word2Vec, which may not have provided sufficient feature richness for the MLP to effectively separate the classes. Additionally, the model's architecture, solver, or activation function could have been better tuned to handle this type of input. Unfortunately, computational constraints posed a significant limitation. The computer used for the project experienced memory issues, restricting the ability to experiment with alternative settings, such as increasing hidden layer sizes, switching solvers, or testing different activation functions. These adjustments might have improved the results for Word2Vec embeddings, underscoring the importance of computational resources when working with more complex models and embeddings.

PTLM

	precision	recall	f1-score	support
Bearish	0.55	0.26	0.36	140
Bullish	0.48	0.48	0.48	176
Neutral	0.81	0.90	0.85	639
accuracy			0.73	955
macro avg	0.61	0.55	0.56	955
weighted avg	0.71	0.73	0.71	955

Pre-trained LLM

The pre-trained tweet sentiment Large Language Model demonstrated moderate performance. The model achieved an accuracy of 0.66, with a precision of 0.63 and a recall of 0.66, culminating in a weighted average f1 score of 0.63. While these results indicate some effectiveness in sentiment classification, they fall short of the performance achieved by simpler embeddings paired with traditional classifiers like Logistic Regression or Naive Bayes.

	precision	recall	f1-score	support
Bearish	0.48	0.29	0.36	140
Bullish	0.48	0.23	0.31	176
Neutral	0.70	0.86	0.78	639
accuracy			0.66	955
macro avg	0.55	0.46	0.48	955
weighted avg	0.63	0.66	0.63	955

One potential factor limiting the LLM's performance is the data cleaning process applied before feeding the tweets into the model. The pre-trained LLM was trained on raw Twitter data, which may retain elements such as URLs, punctuation, and stopwords, contributing to its internal understanding of sentiment patterns. By removing these elements during preprocessing, the input data may have diverged from the format the LLM was optimized for, potentially reducing its ability to leverage its pre-trained weights effectively.

Long Short Term Memory Neural Networks

When applied to the LSTM neural network, the transformer-based pre-trained embedding model outperformed Word2Vec across all metrics in the classification report. The transformer model delivered a final weighted average f1 score of 0.74, showcasing its effectiveness in capturing contextual relationships within the data.

LSTM - PTLM

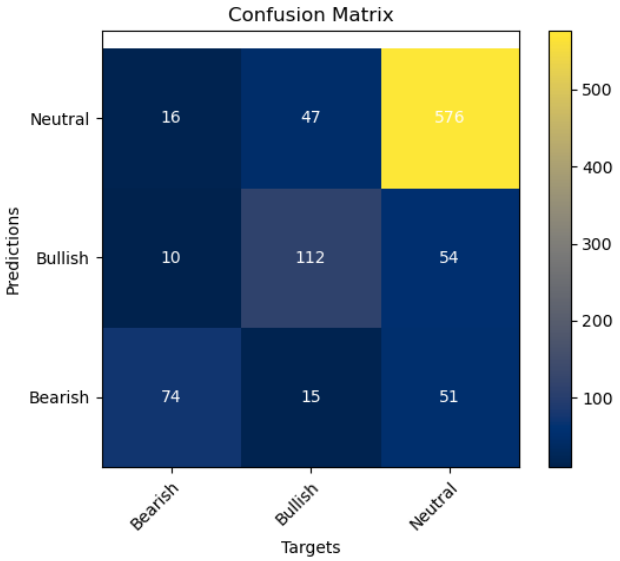
	precision	recall	f1-score	support
Bearish	0.51	0.50	0.51	140
Bullish	0.55	0.59	0.57	176
Neutral	0.85	0.83	0.84	639
accuracy			0.74	955
macro avg	0.64	0.64	0.64	955
weighted avg	0.74	0.74	0.74	955

The superior performance of the transformer model can be attributed to its rich, pre-trained embeddings, which encode extensive semantic and syntactic information. This allows the LSTM to leverage a more meaningful representation of the input data, improving its ability to model long-term dependencies and classify sentiment accurately. In contrast, Word2Vec embeddings, while effective in capturing word-level similarities, lack the context-awareness and complexity of transformer embeddings, leading to comparatively weaker results in this setting. The transformer model's ability to consistently provide better scores across precision, recall, and f1 metrics further highlights its suitability for tasks requiring a nuanced understanding of textual data, particularly when combined with a powerful sequence-based architecture like LSTM.

Conclusion

This extensive analysis thus demonstrates the potential of embedding methods to influence model performance, particularly for imbalanced datasets, and reaffirms the value of choosing methods that align with the complexity and assumptions of the classification algorithm. Furthermore, grid search techniques could have been employed to meticulously find the optimal parameters for models where we can control the hyper parameters, such as the number of neighbors and distance metric in KNN, or a magnitude of parameters in MLP, such as hidden layer size, activation function, solver, learning rate, and batch size. Unfortunately, as mentioned above, the limitations on computer capabilities made it impossible to run these highly demanding algorithms

Balancing all the best performers across the different classification models, the Bag of Words approach combined with the Logistic Regression yielded the best overall results, amounting to an f1 score of 0.79. The confusion matrix shows us in a more visually appealing way how the model predictions fared up to the actual values.



Using this best model, the test data was cleaned, fitted to the training model, predicted and saved, to be compared with the actual labels – unavailable at the time of writing.