NOVA IMS Information Management School

MDSAA

Master's Degree Program in
Data Science and Advanced Analytics

Text Mining

Stock Sentiment: Predicting Market Behavior from Tweets

Chloé Deschanel: 20240693
Diogo Carvalho: 20240694
Ingrid Lopez: 20240692
Rúben Marques: 20240352

Group 05

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa
June, 2025

Table of Contents

I) Introduction

In recent years, the rise of retail investing and the influence of real-time information have transformed how financial markets respond to public sentiment. Among the various data sources that reflect investor mood, Twitter has emerged as a valuable resource due to its immediacy, reach, and volume of financial discussions. However, extracting meaningful insights from noisy and informal tweet content remains a challenge that requires robust natural language processing (NLP) techniques.

This project focuses on the task of stock sentiment classification, where the goal is to categorize financial tweets into three sentiment classes: Bearish, Bullish, and Neutral. The dataset consists of short-form tweets labeled according to their perceived market sentiment. Our objective was to preprocess and vectorize these tweets using both classical and modern feature engineering techniques, and then evaluate the performance of a variety of machine learning and deep learning models—including state-of-the-art transformer-based architectures.

By combining traditional approaches like TF-IDF and Word2Vec with more recent methods such as DistilBERT and financial-domain Sentence-BERT, we sought to understand how different representations and models affect classification performance. Ultimately, the project aims to provide a robust pipeline for financial sentiment prediction, which can support downstream applications in algorithmic trading, market monitoring, and investment decision-making.

II) Data Exploration

For this project, we were given two datasets: a train set and a test set. The train set contains 9543 rows with two columns, text and label, while the test set has 2388 rows, also with two columns, id and text. The label column in the training set includes only three values: 0 for Bearish tweets, 1 for Bullish tweets, and 2 for Neutral tweets. There are no missing values or duplicate rows in either dataset.

Looking at the distribution of sentiment labels in the training data, we found that the average label value is around 1.50. Since these are categorical values, the average doesn't correspond to a real sentiment, but it suggests there's a bit more positive/neutral sentiment overall. From a count perspective (see fig. 1), Bearish (0) account for 1442 tweets (15.1% of the dataset), Bullish (1) account for 1923 tweets (20.2% of the dataset), and Neutral (2) account for the majority with 6178 tweets (64.7% of the dataset). This imbalance clearly shows that most tweets are labeled as Neutral, and Bearish tweets are the least common. This could potentially affect model performance and may need to be addressed later.

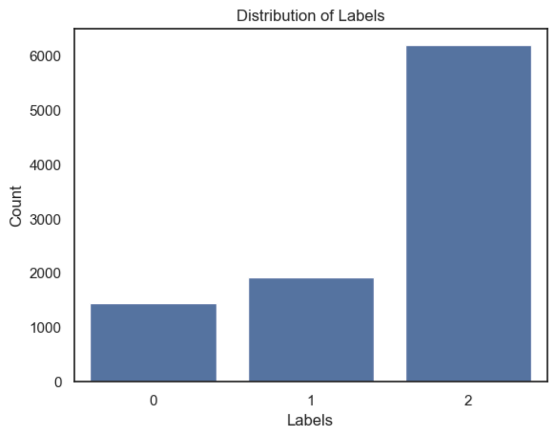


Fig. 1: Distribution of Labels

We then examined the tweet texts in more depth (see fig.2). The average tweet length is about 86 characters or 12 words. Around 75% of the tweets are under 120 characters and contain fewer than 15 words, which confirms that the dataset matches typical tweet structure and is suitable for short-text models like LSTMs or Transformer-based encoders. The shortest tweet is just 2 characters, while the longest has 190 characters or 32 words. Common words in the dataset include high-frequency stop words like "to", "the", "of", and non-word tokens like "-" and "…". These add noise and need to be handled during preprocessing.

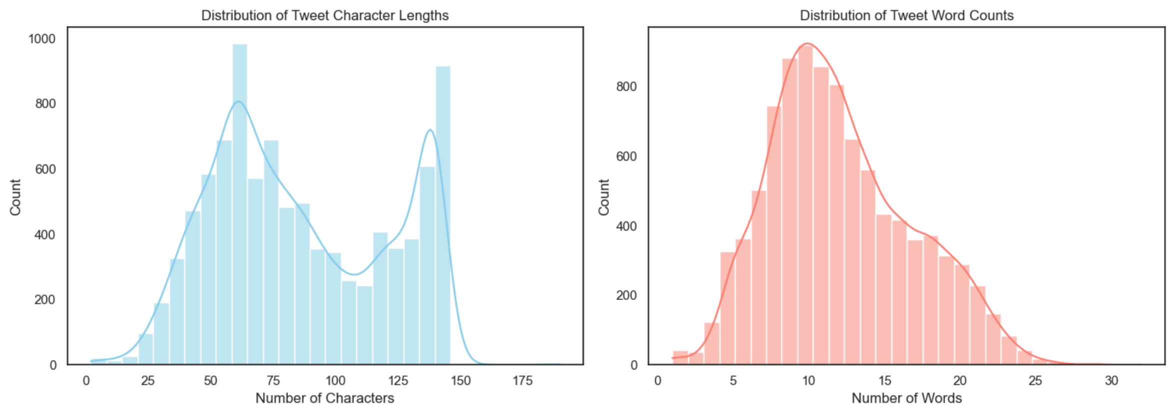


Fig. 2: Distribution of Tweet Character Lengths and Word Counts

Looking at word count by sentiment (see fig. 3), we observed that Neutral tweets tend to be a bit longer than Bearish or Bullish ones. Their word distributions also show more variability, with a wider interquartile range and more extreme outliers.

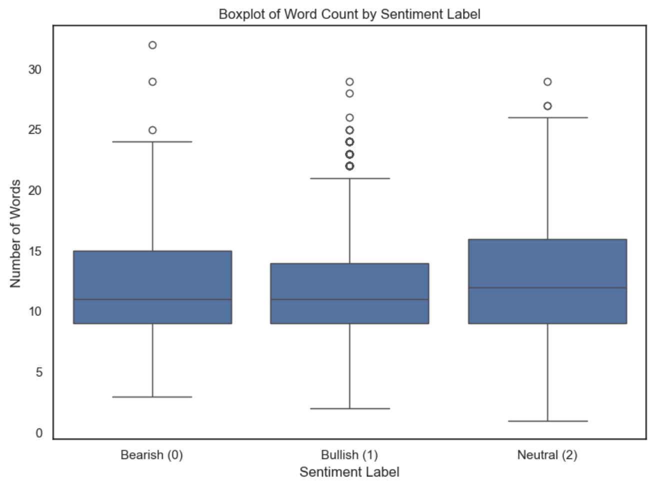
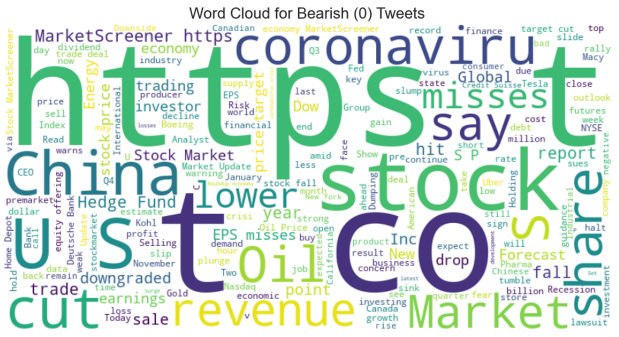
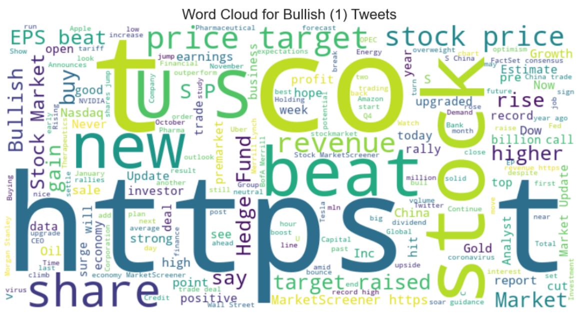


Fig. 3: Boxplot of Word Count by Sentiment Label

When analyzing the vocabulary associated with each label, we noticed distinct patterns (see fig. 4, 5 & 6). Bearish (0) tweets contain more negative and risk-related terms like *cut, misses, lower, decline*, as well as crisis-driven topics like *coronavirus, oil, and China*. Bullish (1) tweets are filled with optimistic signals, like *beat, rise, gain, target raised, soar,* and *buy*, clearly reflecting positive investor sentiment. Neutral (2) tweets are more factual or descriptive, featuring words such as *update, conference call, results, CEO, dividend,* and *MarketScreener.* These likely report earnings, management decisions, or company updates without expressing an opinion.

Interestingly, all classes frequently include noisy tokens like https, co, or rt, indicating a need for proper text cleaning before model training.





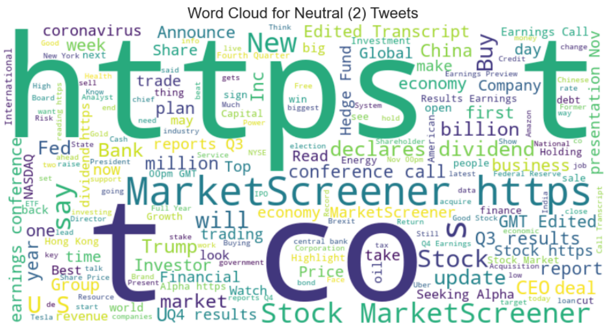


Fig. 4, 5, & 6: Word Cloud for Bearish, Bullish and Neutral Tweets

Lastly, we extracted frequent bigrams to identify common phrase patterns. Some of these reflect syndicated content or automatic link sharing, like *marketscreener https*, *stock marketscreener* and *economy marketscreener*. Others are more domain-specific and relevant for financial sentiment, like *hedge funds, price target, stock price, q3 results* and *earnings conference*. Many of these are likely neutral in tone but still useful for context modeling.

III) Data Preprocessing

To prepare the dataset for training and evaluation, we performed an 80/20 split on the labeled training data. This means 80% of the tweets were used to train the models, and the remaining 20% were reserved for validation. The split was done using stratified sampling, so the distribution of sentiment labels (0: Bearish, 1: Bullish, 2: Neutral) was preserved in both sets. This helps avoid bias and ensures that the model learns from a representative sample of each class. We did not create an additional test set, since a separate unlabeled test file was already provided as part of the project.

Before modeling, we applied a series of preprocessing steps to clean and standardize the tweet text to the train, validation and test sets. The goal was to reduce noise, normalize linguistic forms, and make the data more useful for vectorization and classification. All techniques were chosen based on course guidelines and implemented using NLTK, re, and string. The main steps we applied are:

- **Lowercasing:** Every tweet was converted to lowercase to avoid treating words like "Market" and "market" as different tokens.

- **Noise Removal (Regex-based):** We used regular expressions to strip away irrelevant elements, including URLs (http... or www...), mentions (@), hashtags, retweet markers (RT), and HTML line breaks (<br> or just br).
- **Punctuation and Digit Removal:** All punctuation marks and numeric characters were removed to simplify the vocabulary and reduce dimensionality. This also helps focus the model on actual words rather than symbols or codes.
- **Tokenization:** We used NLTK's TreebankWordTokenizer, which tokenizes based on a standard set of English grammar rules. It doesn't require external downloads like punkt, which makes it more portable.
- **Stopword Removal and Lemmatization:** Common stopwords (e.g., "and", "the", "of") were removed. Then, for the remaining tokens, we applied lemmatization to reduce them to their base forms (e.g., "running" → "run", "companies" → "company"). We also filtered out any token with fewer than 3 characters, since extremely short words often add noise.

This pipeline resulted in cleaner, more compact tweet texts that are better suited for downstream feature engineering and model training. Here's a quick example:

- **Original Tweet:** *"Could Applied DNA Sciences, Inc. (APDN) See a Reversal After Breaking Its 52 Week Low? - The Lamp News"*
- **Cleaned Tweet:** *"could applied dna science inc apdn see reversal eaking week low lamp news"*

However, the typo in "eaking" instead of "breaking" shows how aggressive punctuation removal can occasionally create broken words, which is something to consider in the future.

IV) Feature Engineering

To transform the cleaned tweets into numerical representations suitable for machine learning models, we experimented with five different feature engineering techniques. This helped us understand which representations worked best for classifying sentiment.

a) Bag-of-Words (BoW)

Firstly, we started with the Bag-of-Words (BoW) model, a simple yet effective approach that represents each tweet as a binary vector, indicating the presence (1) or absence (0) of a word in the tweet. We used the CountVectorizer class from sklearn with the binary=True parameter to focus on word occurrence rather than frequency. After fitting the vectorizer to our training data and transforming both train and validation sets, we extracted the most common terms in the corpus, including words like *stock, market,* and *price*.

b) TF-IDF (Term Frequency–Inverse Document Frequency)

Secondly, we used TF-IDF (Term Frequency–Inverse Document Frequency), which penalizes words that appear too frequently across all documents. We used TfidfVectorizer with a max_dfthreshold of 0.8 to ignore very common words and restricted ourselves to unigrams (ngram_range=(1,1)). We inspected the Inverse Document Frequency (IDF) scores to identify rare, domain-specific terms that might carry more weight for classification. A few examples with high IDF values include *firstquaer, fit, fitness, fiscal,* and *fitbit*. These less common terms may not appear often, but when they do, they provide strong contextual clues. That said, some of these (like “firstquaer”) also reveal minor spelling errors or outlier tokens, possibly introduced during preprocessing.

c) Word2Vec

Thirdly, we experimented with Word2Vec, a dense embedding method that captures word meaning based on context. We trained a custom model on our cleaned training data to tailor the embeddings to financial and sentiment-specific language. Each tweet was then represented by the average of its word vectors, producing a fixed-size feature vector (length 100) suitable for traditional classifiers. However, for LSTM models, averaging the vectors wasn’t enough. LSTMs are designed to work with sequences, not fixed-size summaries. To make Word2Vec embeddings compatible with LSTMs, we had to tokenize the text into sequences of word indices, pad those sequences to a fixed length (MAXLEN=100), build an Embedding layer initialized with the pre-trained Word2Vec vectors, and then pass the padded sequences into the Embedding + LSTM layers. This way, the LSTM could leverage the semantic relationships learned by Word2Vec, while also preserving word order, which is crucial for understanding sequence-level context (e.g., “stock falls after earnings” vs. “stock rises after earnings”). To recap, we used Word2Vec in two ways, one as averaged embeddings for traditional ML models and the other as embedding layers for deep learning models like LSTMs.

d) DistilBERT

Next, we tried DistilBERT, a Transformer-based model from Hugging Face that captures rich semantic information and contextual relationships between words. We used DistilBERT in three ways for feature extraction and modeling. In the first approach, we applied DistilBERT’s tokenizer to each tweet, padded/truncated the sequences, and passed them through the pretrained encoder. From the output, we extracted the [CLS] token embedding, a 768-dimensional vector that summarizes the tweet’s meaning. These vectors were then used as input features for traditional classifiers like Logistic Regression and KNN. In the second approach, we extracted token-level embeddings from DistilBERT to leverage sequence modeling with LSTM. Each tweet was padded to a fixed length (e.g., 32 tokens), and the model outputted a 3D tensor: (samples × max_length × 768). This preserved both word order and semantic context, making it well-suited for LSTM layers to capture sentiment flow across the sequence. In the final way, we prepared a pipeline to fine-tune DistilBERT itself using PyTorch. Finally we built two dataset classes, FinDataset for supervised learning (includes tokenized input and label), and InferenceDataset for test-time inference (only tokenized input). Each tweet was tokenized using consistent settings (padding, truncation, max_length=64), and inputs were converted into tensors compatible with Hugging Face’s Trainer or custom training loops. This setup allowed us to fine-tune the DistilBERT model directly on our classification task.

e) Finance-specific Sentence-BERT

Finally, we used a finance-specific Sentence-BERT model. This model builds on the original SBERT architecture, but is fine-tuned on financial texts, making it especially suited for the kind of data we’re working with. Unlike standard BERT, which requires additional pooling steps, SBERT directly outputs a dense, sentence-level embedding optimized for semantic similarity tasks. This made it easy to apply to our tweet-level sentiment classification. Each tweet was converted into a 384-dimensional embedding, capturing rich contextual and financial-domain knowledge.

V) Classification Models

We used various machine learning and deep learning models to classify the sentiment of the tweets into one of three categories: Bearish, Bullish, or Neutral. Each model was trained and evaluated using the different text processing methods presented in the previous section.

a) K-Nearest Neighbors (KNN)

As a starting point for the sentiment classification of tweets, we first use the K-Nearest Neighbours (KNN) algorithm. By examining the most frequent label among its k nearest neighbours in the training set, this model predicts the label of a new tweet. In the first tests we used k = 5. This model was applied to five different types of input features:

- Bag-of-Words (BoW)
- TF-IDF
- Word2Vec embeddings
- DistilBERT embeddings
- Sentence-BERT embeddings

Results differed depending on the representation. Sentence-BERT embeddings showed somewhat more balanced behaviour across sentiment classes, while simpler vectorisations, such as BoW and TF-IDF, produced fairly modest results. DistilBERT embeddings yielded the most promising performance configuration, which was then refined by grid search. Numerous combinations of weighting schemes, distance metrics and neighbour counts were tested during this tuning process. The best configuration improved F1 scores in the validation set using three neighbours, cosine distance and distance-based weighting. As the most robust KNN configuration, this adjusted version was used as a standard for evaluating subsequent more complex models.

b) Logistic Regression

We then used logistic regression on the same five representations: Sentence-BERT, DistilBERT, Word2Vec, TF-IDF and BoW. This model, which is often used for text classification tasks, works especially well with high-dimensional sparse data, such as TF-IDF or BoW vectors. The results differed depending on the input:

- BoW and TF-IDF both produced consistently high validation scores. Even before tuning, BoW produced highly competitive results.
- Of all the representations tested, Word2Vec performed the worst, indicating that these embeddings may not work well with linear classifiers.
- DistilBERT and Sentence-BERT embeddings offered richer semantic representations and resulted in stable, mid-range scores.

To further improve performance, we fine-tuned the best-performing version of the model using grid search.

c) Multi-Layer Perceptron (MLP)

We then used a multilayer perceptron (MLP), a neural network model capable of capturing nonlinear interactions within the feature space. All five vector representations were used to train the model.

- Overall, BoW and TF-IDF obtained the best results, demonstrating that MLPs can learn efficiently from sparse lexical representations.
- In contrast, Word2Vec showed inferior performance, probably because the limited depth of the model was insufficient to capture the underlying semantic structure of the distributed embeddings.
- The Sentence-BERT and DistilBERT representations also performed well, but did not significantly improve on the simpler alternatives.

The model was then fitted using TF-IDF inputs, which had already shown promise. Following a hyperparameter search, marginal improvements in validation scores were obtained, confirming that the MLP was already operating close to its capability with TF-IDF features.

d) LSTM

To further enhance sentiment classification, we implemented a Long Short-Term Memory (LSTM) neural network, which can effectively model the sequential nature of text data. The LSTM was tested using three types of input features:

- Word2Vec embeddings (as sequences)
- Transformer BERT embeddings (as token-level sequences)
- Sequence BERT embeddings (reshaped from sentence embeddings)

The LSTM achieved clearly better results with sequential Word2Vec inputs compared to traditional machine learning models that used averaged embeddings, demonstrating its ability to leverage word order. The strongest and most consistent performance was observed when the LSTM was combined with token-level transformer embeddings, as this approach allowed the model to capture rich contextual and sequential information, resulting in the most robust classification across sentiment classes. Sequence BERT embeddings, when adapted to pseudo-sequences, led to moderate improvements, though the benefits were not as pronounced as with token-level transformer inputs. Throughout development, we applied techniques such as focal loss and regularization to address class imbalance and overfitting. Overall, the LSTM outperformed earlier models based on classical representations and served as a strong deep learning baseline for sentiment classification in this task.

e) Transformer

As a final approach for sentiment classification of tweets, we applied transformer-based models by fine-tuning large pre-trained neural networks on our data. In these experiments, each tweet was processed as a sequence of tokens using two different transformer architectures:

- Fine-tuned DistilBERT
- Fine-tuned FinancialBERT

Both models were adapted to the sentiment classification task by updating their parameters with our labeled training set. The use of self-attention layers within these models allows them to capture contextual information and complex relationships within each tweet. The fine-tuning process followed standard procedures, training the models for a fixed number of epochs using cross-entropy loss and monitoring progress on a validation split. This transformer-based configuration serves as a reference for state-of-the-art natural language understanding in the context of sentiment analysis, and provides a benchmark for comparison with classical and deep learning methods explored earlier.

VI) Evaluation & Results

To assess model performance, we focused on three standard classification metrics: Precision, Recall, and F1-Score, computed on the validation set. Given the class imbalance in the dataset, where Neutral tweets significantly outnumber Bearish and Bullish ones, we emphasized macro-averaged F1-score as our primary evaluation metric to ensure fair performance across all classes. The table in the appendix summarizes the validation results across all model-embedding combinations. Our key observations are as follows:

- Classical Models with Sparse Representations: Logistic Regression and MLP trained on TF-IDF and BoW performed surprisingly well, with F1-scores reaching ~0.69. These models proved effective in capturing lexical patterns despite their simplicity and computational efficiency.
- Dense Embeddings and Word2Vec: Models trained on averaged Word2Vec embeddings generally underperformed, particularly with linear classifiers. However, when combined with LSTM and used as sequential inputs, Word2Vec yielded improved results (~0.55 F1), showcasing the importance of sequence modeling for contextual embeddings.
- Transformer Embeddings and Fine-tuning: Models leveraging DistilBERT and Sentence-BERT achieved solid mid-range results when used as static embeddings. However, the best overall performance was achieved through fine-tuning DistilBERT, which led to a macro-averaged F1-score of **0.7485**, outperforming all previous configurations.

This result highlights the strength of transformer-based models fine-tuned for specific tasks, especially in capturing the nuances of financial language. The use of contextualized word embeddings, combined with end-to-end optimization, allowed the model to learn fine-grained distinctions between sentiment categories. In conclusion, while traditional models offer good baselines, the best performance was achieved through fine-tuning a pre-trained transformer, confirming its value in financial text classification tasks.

VII) Conclusion

This project explored the use of natural language processing techniques to classify financial tweets by sentiment, Bearish, Bullish, or Neutral, based on short text inputs. Through a comprehensive pipeline that included data exploration, text preprocessing, feature engineering, and model evaluation, we were able to benchmark a range of machine learning and deep learning approaches for this task. Our findings show that while classical models like Logistic Regression and MLP perform well with TF-IDF and Bag-of-Words representations, their performance is ultimately surpassed by transformer-based models. Specifically, fine-tuning DistilBERT on our labeled data led to the highest validation performance, achieving a macro-averaged F1-score of 0.7485. This underscores the advantage of contextualized embeddings and transfer learning for sentiment classification in finance. Beyond model performance, the project highlighted the importance of thoughtful preprocessing, such as lemmatization, stopword removal, and cleaning domain-specific noise, and the trade-offs between interpretability and complexity. It also emphasized how class imbalance and vocabulary noise can influence model behavior.

Overall, this work demonstrates that state-of-the-art transformer architectures can be successfully adapted to short-form financial text, providing accurate and scalable solutions for market sentiment analysis.

VIII) Appendix: Validation Metrics

Model	**Representation**	**Val Precision**	**Val Recall**	**Val F1-Score**
KNN	BoW	0.683126	0.452359	0.469916
KNN	TF-IDF	0.680698	0.400908	0.391255
KNN	Word2Vec	0.435660	0.418572	0.423317
KNN	Distil BERT	0.557215	0.561505	0.559061
KNN	Sentence BERT	0.521778	0.515665	0.518512
KNN (CV-GS)	Transformer BERT	0.586272	0.560582	0.569738
Logistic R.	BoW	0.750979	0.652088	0.687466
Logistic R.	TF-IDF	0.765356	0.594114	0.638318
Logistic R.	Word2Vec	0.370804	0.357817	0.315742
Logistic R.	Distil BERT	0.675642	0.595069	0.622866
Logistic R.	Sequence BERT	0.557526	0.506367	0.521700
Logistic R. (CV-GS)	BoW	0.751196	0.652663	0.687906
MLP	BoW	0.736730	0.656683	0.685633
MLP	TF-IDF	0.722293	0.672801	0.692368
MLP	Word2Vec	0.368498	0.398722	0.372375
MLP	Distil BERT	0.707471	0.578559	0.616092
MLP	Sequence BERT	0.586619	0.493145	0.513628
MLP (CV-GS)	TF-IDF	0.722293	0.672801	0.692368
LSTM	Word2Vec	0.705456	0.555200	0.552092
LSTM	Transformer BERT	0.598871	0.655484	0.615334
LSTM	Sequence BERT	0.514047	0.529220	0.519753
Transformer	Fine-tuned DistilBERT	0.762247	0.737341	0.748515
Transformer	Fine-tuned Financial BERT	0.636303	0.580315	0.600140