FITE4801 Final Year Project Interim Report

Sentiment Analysis for Finance News Headlines

Kwok Ka Tin (UID: 3035684843)

17th January 2024

Abstract

Computer experts have been using artificial intelligence (AI) in the finance sector for investments since the 1990s. While stock price prediction based on historical price data remains one of the more popular ways to use AI, the use of Natural Language Processing (NLP) for sentiment analysis on financial text such as financial news headlines has gained traction. Models like FinBERT are developed for this purpose. However, most of these models are unable to analyse the sentiment of the different entities mentioned in the headline. This paper proposes the development of a new model that can analyse the sentiment of the different entities separately. The data used were from the datasets FiQA Task 1 and SEntFiN 1.0. There are three types of candidate classification algorithms that this project may choose from: Scikit-Learn classifiers, Long Short-Term Memory (LSTM), and XLNet. The candidate models were developed after the data goes through data preprocessing steps and a train-test split. The models were evaluated by their accuracy and F1 score. For the first experiment, the Logistic Regression model from Scikit-Learn is the best performing model, with an accuracy of 76%. The poor performances of the LSTM and XLNet models can be attributed to the lack of finetuning, the lack of understanding, and flawed word embedding methods. Therefore, by the end of February, solutions to tackle these three problems will be executed to improve the performances of the two models.

Table of Contents

List of Tables

List of Abbreviations

Abbreviations	**Definition**
AI	Artificial Intelligence
NLP	Natural Language Processing
BERT	Bidirectional Encoder Representations from Transformers
AE	Autoencoding
AR	Autoregressive
MLM	Masked Language Model
LLM	Large Language Model
FiQA	Financial Opinion Mining and Question Answering
LSTM	Long Short-Term Memory
POS	Part of Speech
NER	Named-Entity Recognition
BoW	Bag of Words
SVC	Support Vector Classification

1. Introduction

Ever since the 1990s, computer experts have come up with ways to make use of artificial intelligence (AI) for trading financial instruments in global markets [1]. Using AI for trading can improve accuracy and efficiency as it can analyse large amounts of ever-changing market information in real-time in minutes, replacing the old method of manual research which can take weeks [2]. However, trading with AI is not without its weaknesses. Since AI is heavily dependent on the training data it receives. When the historical training data does not fit with the current market conditions, AI cannot predict the trends correctly and produce outdated decisions [2]. Since past results can only reflect how the market reacts to previous external events, it cannot ensure how it would behave under current conditions and events.

Therefore, instead of predicting trends by analysing past data, predicting market reactions by analysing current events seems to be a better idea. Current events are often expressed textually in financial news or reviews. To analyse these textual data, sentiment analysis or opinion mining can be used. Sentiment analysis is the method of classifying the opinions of an author, usually either positive or negative, by analysing the texts the author wrote [3]. If the news expresses positive attitudes towards a stock, it can raise the collective confidence of investors, which will lead to better performance of the stock [4].

While any individual investor or financial analyst can scrutinise finance news and reviews and classify them by their “positiveness”, going through the vast amount of news and reviews published by different media outlets and stock companies would be a strenuous task. Besides, human classification will risk human bias and ineptness if the analyst is inexperienced [5]. Therefore, AI, which as previously mentioned can predict with better accuracy and efficiency, can also be adopted here. This can be achieved by Natural Language Processing (NLP), which is a combination of AI and computational linguistics, to let computers read and understand human language and textual data [6]. This project aims to create an NLP model that can predict the trend of a certain stock by performing computational sentiment analysis on financial news headlines.

The rest of this progress report is structured as follows: Section 2 discusses previous attempts at applying sentiment analysis and artificial intelligence to the finance sector. Section 3 describes the proposed designs and methods that will be used to develop the sentiment analysis program. Section 4 suggests some expected results of the proposed methodology and discusses

some foreseeable challenges during development. Finally, Section 5 concludes this progress report with a summary of the entire report.

2. Literature Review

"FinBERT" is an existing sentiment analysis model developed for financial news sentiment analysis. It is developed by Araci and it is based on a language model named BERT (Bidirectional Encoder Representations from Transformers) [7]. BERT is an autoencoding (AE) language analysis model. As its name suggests, it can comprehend text bidirectionally as compared to autoregressive (AR) language models like GPT (Generative Pre-trained Transformer), which can only read text unidirectionally [8]. BERT achieves its bidirectionality by using a masked language model (MLM) that replaces some words from the input with a masked token. The model then attempts to predict the masked words based on the words around them [9]. When FinBERT was published in 2019, it claimed that it outperformed the best NLP models by a significant margin at that time. However, it has been 4 years since FinBERT was introduced and new language models were also introduced during these years, such XLNet, which is said to outperform BERT in several text classification tasks, including sentiment analysis [10]. BERT is also revealed to have disadvantages. The use of MLM corrupts input data and information will be lost. BERT also assumes all masked tokens are independent, which might be problematic if many of the important words are masked [11].

Moreover, FinBERT is unable to output sentiments based on the different entities in the text. For instance, for the news headline "Nikkei rises as Yen plumbs 6 years low", FinBERT outputs "Negative". However, Nikkei, which is said to be rising, should have a positive sentiment. Similar headlines which mention multiple entities with polarizing sentiment appears often. Outputting only one sentiment of one entity is inefficient and will easily cause confusion to users. Therefore, this project aims to fix this problem of FinBERT and create a model that can output a different sentiment for the different entities in the headline.

3. Proposed Methodology

This section elaborates on the proposed methodology of this project, starting brief explanation of the outline of the program. Next, the datasets, the classification algorithms, and the language and libraries that will be used and experimented with will be introduced. The data preprocessing stage will also be elaborated. Finally, the evaluation metrics for the sentiment analysis models will be covered.

3.1 Outline of the program

The main idea of the program is to take a financial news headline as input. When the program receives an input, it first cleans the text and removes noises and meaningless parts. Next, it will recognize entities or organizations from the headline and extract them. For example, the program should be able to find “Nikkei” and “Yen” from the headline “Nikkei rises as Yen plumbs 6 years low”. This extraction is planned to be done by a pre-trained Named-Entity Recognition (NER) model that can recognize proper nouns such as people and organizations in a text effectively. After extracting, the headline and the name of entity pair will go through another text preprocessing step which turns the text into computer-readable format, or number vectors. The number vectors will be put into a trained language model. The model studies the the entity and the headline and makes a prediction of the sentiment of the sentence towards the entity. This prediction will be outputted to the user as the output of the program.

3.2 Datasets

Datasets are the relevant data used to develop machine learning models and artificial intelligence. The model will create rules and find relationships based on the training datasets which will be used for predictions. For this project, two datasets will be used. Both datasets include financial news headlines, the entities mentioned in the headline, and the sentiment of the headline to the corresponding entity.

3.2.1 FiQA Task 1

FiQA (Financial Opinion Mining and Question Answering) is an open challenge organized by The Web Conference held in Lyon in 2018. The challenge was split into 2 tasks, Task 1 asked participants to submit a system that can predict a sentiment score for a given financial text, which is similar to the objective of this paper. This dataset scores the sentences with a value between -1 and 1, with -1 being the most negative sentiment and vice versa. Apart from the sentence and the sentiment score, this dataset also includes a “target” which is the company or stock the sentence is referring to [12]. This dataset is chosen also for its credibility. The challenge is part of a long-running, professional conference series [13]. The organizers of this challenge are researchers and professors from credible universities and the European Commission, which runs the European Union. Therefore, this dataset has a solid foundation and can be trusted to be used as a reliable data source.

3.2.2 SEntFiN 1.0

SEntFiN 1.0 is a dataset created by Sinha, Kedas, and Kumar for the paper "SEntFiN 1.0: Entity-Aware Sentiment Analysis for Financial News". There are 10753 news headlines in this dataset, with around 2800 of the headlines containing multiple entities. Since some headlines with multiple entities may cause opposite sentiments to the entities, each entity is individually annotated. There are overall 14404 sentiment annotations in this dataset, with 35.23% of the entities receiving a positive sentiment, 26.48% negative, and 39.29% neutral [14]. The detailed process of creating the dataset was published in a journal and the annotators also had background in finance. Also, the indication of the entities of each sentence is helpful for the latter stages. Moreover, the close percentages between positive, negative, and neutral sentiment labels are appreciated.

3.3 Classification Algorithms

There are multiple machine-learning algorithms for classification tasks. Many of which are suitable for NLP and sentiment analysis. This project picked three kinds of the algorithms to experiment with to see which performs the best. The three kinds of algorithms are Scikit-Learn classifiers, Long Short-Term Memory (LSTM), and XLNet.

3.3.1 Scikit-Learn Classifiers

The Scikit-Learn library offers multiple classification models. These models are easy to implement and has active community support. Therefore, it is commonly used in many different areas. The library also provides functions such as Grid Search and Cross Validation, for evaluating models and selecting the best hyperparameters [15]. Therefore, some classification models from this library will be used in this project. Due to the simplicity of the models, they are used as benchmarks for the other two deep learning models.

This project evaluated 4 models from the library: Logistic Regression, Linear Support Vector Classification (SVC), Naïve Bayes, and Random Forest. These four models were evaluated with a 5-fold cross validation and the two best performing models will be used to compare with the other two models.

3.3.2 LSTM

LSTM is an architecture of recurrent neural network (RNN) which is a deep learning algorithm. LSTM is designed for modelling sequential data. LSTM overcomes the vanishing gradient problem which is prevalent in RNNs. For an RNN, the neurons are connected with time as layers. When building an RNN, the weights of a neuron are calculated and optimized by the

gradient descent method, which involves backpropagation, or multiplying back the previous neuron weights in the network. As the network becomes deeper, the updated gradients have to backpropagate through more layers to reach the first layer. During this process, the updated gradients will be multiplied by neuron weights repeatedly, which diminishes the gradient. A gradient too small cannot carry new information and update the weights in the neurons, which is undesired. LSTM solves this problem by assigning a memory cell which helps remember information back in time, making it optimal for sequential data [16]. Textual data can be considered sequential. Therefore, LSTM, which can overcome this problem, will perform well in NLP and sentiment analysis [17].

3.3.3 XLNet

XLNet is a pre-trained NLP model that uses AR modelling. As mentioned, traditional AR models can only read text from one direction, and AE models like BERT can corrupt input data. XLNet is designed to solve the disadvantages of both AR and AE. It can read contexts bidirectionally without corrupting any important input data. This is executed by a permutation language model which allows bidirectional understanding by permutating the words in the text. As previously mentioned, XLNet boasts it outruns BERT on 20 tasks, including sentiment analysis [10]. Since FinBERT is already developed, it is hoped that a similar model built with XLNet can also outperform the FinBERT in the task.

3.4 Languages and Libraries

As a programming project, it is important to select an ideal programming language to complete to task. Python was selected since it is the most popular programming language in machine learning, with over half of the machine learning programmers using the language, surpassing other languages specifically developed for machine learning such as R and Julia [18]. The popularity of Python can be accounted for its simple and intuitive syntax, as well as its extensibility that can be integrated with different platforms and languages. Most importantly, Python provides easy access to many different libraries and frameworks that can be used to perform various tasks, including machine learning and text analysis [19]. For this project, the libraries Scikit-Learn, Keras, and Transformers will be imported for accessing the models of Naïve Bayes, LSTM, and XLNet respectively. Pandas, SpaCy, and Textacy will be used for the data preprocessing stage.

3.5 Data Preprocessing

"Garbage in, garbage out" is an important concept in computer science. It means if we input bad-quality data into a system, the system will only output bad-quality results [20]. Therefore, to ensure a quality prediction model, the input data used for training the model must not be garbage. This is why the data preprocessing stage is a crucial stage in developing an artificial intelligence model. For preprocessing textual data for this project, a number of stages is required before training the model. These stages are elaborated below

3.5.1 Standardizing Labels

This step is essential for this project since this project uses two datasets which labelled their sentence sentiment differently. To standardize all datasets, positive sentiments will be labelled 2, negative sentiments will be labelled 0, and neutral statements will be labelled 1. For FiQA Task 1, which uses float numbers as sentiment scores, all negative sentiment scores will be generalized to 0 and vice versa. This rule is chosen because classification models often output an array of percentages of the certainty of the labels. Since arrays are indexed with integers starting from 0, the percentages of the array outputted will follow the sequence: negative, neutral, and positive. If the percentage at index 0 is the greatest, the model predicts a negative sentiment and so on.

3.5.2 Data Cleaning: Identifying and Removing Noises

For text analysis, noises refer to the parts in a sentence that serve no value in the analysis and prediction such as HTML tags, sequences of meaningless symbols, and URLs. These noises provide no extra meaning to the sentence and therefore, will confuse the algorithm and lead to poor performance if they remain in the input data. While it is unlikely for noises to appear in proper news headlines, it is good practice to search for them just in case there are errors in the datasets when importing.

3.5.3 Data Cleaning: Character Normalization

This step is to standardize the characters or symbols in a text into ASCII formats. Humans can understand how characters with accents or umlauts, such as ć or ä, are sometimes replaced by their unaccented version. But computers cannot and will treat the accented characters and the unaccented counterparts as two completely different letters. This problem also persists for symbols such as all different types of quotation marks. To standardize all these characters and symbols, Textacy will be used. It has a preprocessing module that can normalize accented characters and quotation marks, which makes this step simpler.

3.5.4 Word Tokenization

Tokenization is an important step when preprocessing data for NLP. In this step, the sentences will be broken down into individual words known as tokens. By separating each word from a complete sentence into tokens, the algorithm can understand the context of each token more easily and analyse the text better [21]. This step is planned to be carried out by using the SpaCy library.

3.5.5 Text Vectorization

After tokenization each word, the tokens will be transformed into number vectors. This step is essential in NLP tasks because most computer models are not capable of interpreting words and plain text like humans do. They are only great at calculating numbers. Therefore, before building a computer model, the textual data needs to be turned into numerical form. To do so, each token will be mapped into a number vector using a dictionary of words. This process is called Text Vectorization or Word Embedding [22].

This project uses the Count Vectorizer function from Scikit-Learn to perform word vectorization. This function makes use of a model called Bag-of-Words (BoW), which creates a vocabulary of all words in the column and create a binary vector for each sentence based on the presence of words in the sentence [23]. Since the Count Vectorizer can only be create a BoW for one column at a time, the two columns of sentence and entity will have to be concatenated before applying Count Vectorizer on the combined column.

3.5.6 Train-Test Split

After the extraction, the data frame is then split into 80% and 20% for the train-split, which means that 80% of the total data will be used to train the data and the remaining 20% of the data will be used for testing. Testing is where the models make predictions after establishing rules from the training data and the prediction is compared to the actual label. This step is important for evaluating the models to see which has the best performance.

3.6 Evaluation Methods

To compare the performances of the algorithms, the accuracy and F1 score of the results will be calculated. Accuracy is the percentage of correctly predicted classes out of all predictions. It is the most basic metric and works well even for multi-class classifications like this project. The F1 score calculates the harmonic mean of the other common evaluation metrics Precision and Recall [24]. Precision is the percentage of True Positives out of the sum of True Positives and False Positives. True Positives are the cases predicted positive and are positive, while False

Positives are cases predicted positive but are negative. Recall is the percentage of True Positive cases out of the sum of cases that are positives [25]. Since this project is a multi-class classification, the weighted F1 score will be calculated. The weighted F1 score is the weighted average of the F1 score of all three labels “0”, “1”, and “2”.

4. Model Experiment and Results

For the experiment, A cross-validation test was first performed on the 4 candidate classification algorithms from Scikit-Learn with the SEntFiN 1.0 dataset. This is to find the algorithms with the best accuracy. After which, simple models were built with the SEntFiN 1.0 dataset using the two algorithms with the best accuracy in the cross-validation test, as well as LSTM and XLNet. The accuracies and the F1 score were recorded and compared.

4.1 Cross-Validation Test

The 4 Scikit-Learn classification algorithms were put into a 5-fold cross validation and the mean accuracy and the standard deviation is calculated. The results of the are listed in Table 1.

Model	Mean Accuracy	Mean Weighted F1 Score
Linear SVC	0.676596	0.675541
Logistic Regression	0.701173	0.700146
Multinomial Naïve Bayes	0.662674	0.662134
Random Forest Classifier	0.640605	0.628298

Table 1: Mean accuracy and weighted F1 score of the cross validation of the 4 Scikit-Learn classification algorithms.

The results suggest that Logistic Regression and Linear SVC are the best performing algorithms of the four, with a mean accuracy of around 0.70 and 0.67 respectively. Therefore, the following algorithms will be used in the next stage of the experiment.

4.2 First Results of Models

5. The test accuracy and F1 Score of all 4 models is listed in Table 2.

Model Name	Test Accuracy	Test Weighted F1 Score
Logistic Regression	0.765569	0.762049
Linear SVC	0.743822	0.743814
LSTM	0.747302	0.762049
XLNet	0.390184	0.505370

Table 2: The test accuracies and F1 scores from the primary experiments of the 4 models.

From the first results, it suggests that Logistic Regression has the best performance, with an accuracy of 0.76 and a weighted F1 score of 0.76 as well. The LSTM model came second in this run with an accuracy of 0.74 and a weighted F1 score of 0.76. However, the model performance is unstable, the accuracy ranges from around 0.72 to 0.76. Therefore, its performance is comparable to the Linear SVC model.

XLNet has the worst performance, with only an accuracy of 0.39. On closer inspection, it is revealed that the model only predicted the Neutral label "1" for every input.

6. Results Discussion

The results from the primary experiment suggest that Logistic Regression is the best-performing model. However, all Scikit-Learn models are planned to be benchmarks. They are not planned be adopted in the final model. Therefore, the other two models failed to reach the benchmark. The XLNet is not even a model for predicting the same label for everything. Three possible reasons to the poor performance of the LSTM and XLNet, as well as the solutions, are mentioned below.

6.1 No Finetuning

Finetuning is the process of taking a machine learning model that is already trained and making minor adjustments to its hyperparameters so the model can perform better for a specific task [26]. These pre-trained models are usually trained with a vast and diverse dataset. Therefore, the model is already familiar with data. This results in more efficient training and better results. With finetuning, the model can be trained to be a new model for a specific task. For instance, a pre-trained language model can be used and finetune it into a model for analysing the sentiment of financial news headlines.

For my initial models, I just quickly built a simple LSTM model. While building a model from scratch can also lead to good performances, it requires much effort in tuning the learning rate, dropout size, batch size et cetera. Since I did not tune any of these hyperparameters after the first training, the model failed to reach the benchmark. Therefore, it is recommended that I should try loading a pre-trained LSTM language model and finetuning it to the purpose of this project. It is hoped that with finetuning, the model can be more efficient and accurate.

6.2 Lack of Understanding

The failure of the XLNet model can be attributed to my lack of understanding towards pre-trained models. Since I have no experience in using pre-trained models from Hugging Face or

the Transformers library, I underestimated the effort needed to train the model. This results in the numerous error messages I encountered during the experiment, such as a Value Error about the incompatible shapes of the layers, as well as the poor performance of the final model.

To solve this issue, more research regarding XLNet will be done. This includes the detailed mechanisms and the use cases of the model. I will also read more about the functions of the Transformers library regarding XLNet so I can create a model quicker and encounter fewer errors. Moreover, I will look up some example codes on the Internet to learn how experienced programmers use the model. It is hoped that I can understand more about this model once I finish researching and I can develop a more accurate model more effectively.

6.3 Unideal Text Vectorization Techniques

One highlight of this project is the ability to separate the different entities in a news headline and figure out the sentiment of each entity. Therefore, two features are needed when training the model: the sentence and the entity. Before feeding these textual features into the model, they have to be numericized to become computer readable.

In my attempt, I used a BoW model, which is a very simple word vectorization technique. However, as mentioned, this function can only create a BoW on one feature at a time. To vectorize both the sentence and the entity feature, one solution is to create two separate BoW. However, this is unideal as the entity feature is dependent on the sentence feature. A BoW with only words in the entity feature cannot show the dependency between the entity and the sentence. Therefore, a different solution is used, that is to concatenate the two strings into one and then build a BoW on the combined string. However, this solution is also unable to show the dependency.

To solve this problem, two more solutions are proposed. Firstly, feature extraction can be done on the sentences. The SpaCy library can be used to extract keywords and phrases after the target entity in a sentence. These extracted words and phrases can be put into a new feature and these words will be vectorized instead of the sentence itself. This way, only the words that is relevant to the entity is included, which can emphasize the entity better. Secondly, another more advanced word vectorizer, such as GloVe, can be used. GloVe is a pre-trained word embedding model with vectors for about 6 billion words and some other common symbols [27]. It can also understand semantic meaning of words as the model is trained with word pairs on a co-occurrence matrix [28]. Since GloVe uses global statistics, it is unnecessary to combine the

two columns for the list of vocabulary. These two methods can be tested to see if improving the vectorization of words can improve the performance of the models.

7. Project Schedule

For the second semester, the second phase of the project will be carried out, which is to improving the LSTM and the XLNet models by adopting the solutions mentioned in the previous section. It is planned that the second phase will conclude by the end of February. If the solutions are ineffective and the models still failed to reach the benchmark, different solutions will be devised. Otherwise, the third phase of the project will commence, that is to explore more kinds of classification models, such as other RNNs and pre-trained LLMs. Also, the NER for finding entities before the prediction has to be ensured that it can recognize accurately. Otherwise, the prediction output might be incomplete.

8. Conclusion

In this paper, the development of a sentiment analysis program for financial news headlines powered by AI is proposed. This program aims to analyse the hidden opinions behind a news headline from a financial news which can help investors in choosing what companies to invest in. While some projects have already attempted to develop models for text analysis tasks for the finance industry, such as FinBERT, this project attempts to develop a model that can identify entities and companies in the news headline and analyse the sentiment of each entity, which is not a feature in FinBERT. The proposed method for developing the model is mentioned in this paper. Two credible datasets were used as input data. Three kinds of classification algorithms: Scikit-Learn classification models, LSTM, and XLNet were experimented with to find out the best performer. Python was used for its simplicity and accessibility to different libraries. The process of data preprocessing is elaborated and the final models were evaluated by accuracy and F1 score. The experiment results suggest that the Logistic Regression Classifier from Scikit-Learn has the best performance, with a accuracy of 76% and a F1 score of 0.76. However, since the Scikit-Learn models are meant to be used as a benchmark, it is concluded that the other two more complicated models failed to reach the benchmark. Three possible reasons for the poor performances of the two models are suggested: no finetuning of models, lack of understanding of XLNet, and the suboptimal word embedding techniques. To solve the above problems, experiments about finetuning pre-trained models and new word extraction and embedding techniques, as well as more research about XLNet will be conducted by February.

References

[1] F. G. D. C. Ferreira, A. H. Gandomi, and R. T. N. Cardoso, "Artificial Intelligence Applied to Stock Market Trading: A Review," *IEEE Access,* vol. 9, pp. 30898–30917, Jan. 2021, doi: 10.1109/access.2021.3058133.

[2] A. O. Akin-Davidson, "The Use of Artificial Intelligence in Algorithmic Trading in the Global Market." Jan. 2023. [Online]. Available: https://www.linkedin.com/pulse/use-artificial-intelligence-algorithmic-trading-alfred-olutola

[3] J. M. Brain, "'Past performance is not necessarily indicative of future results' - the proven-in-use argument and the retrospective application of modern standards," *5th IET International Conference on System Safety 2010*, 2010, doi: 10.1049/cp.2010.0833.

[4] Q. Xiao and B. Ihnaini, "Stock trend prediction using sentiment analysis," *PeerJ Comput Sci*, vol. 9, p. e1293, Jan. 2023, doi: 10.7717/peerj-cs.1293.

[5] A. Yadav, C. K. Jha, A. Sharan, and V. Vaish, "Sentiment analysis of financial news using unsupervised approach," *Procedia Comput Sci*, vol. 167, pp. 589–598, 2020, doi: 10.1016/j.procs.2020.03.325.

[6] IBM, "What is Natural Language Processing (NLP)? ." IBM. [Online]. Available: https://www.ibm.com/topics/natural-language-processing

[7] D. Araci, "FinBERT: Financial Sentiment Analysis with Pre-trained Language Models," *arXiv (Cornell University)*, Jan. 2019, doi: 10.48550/arxiv.1908.10063.

[8] M. Xiao, "Understanding Language using XLNet with autoregressive pre-training." Jan. 2020. [Online]. Available: https://medium.com/@zxiao2015/understanding-language-using-xlnet-with-autoregressive-pre-training-9c86e5bea443

[9] "Masked Language Models." Jan. 2023. [Online]. Available: https://saturncloud.io/glossary/masked-language-models

[10] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, vol. 517, pp. 5753–5763, 2019.

[11] A. A. Falaki, “What are the differences in Pre-Trained Transformer-base models like BERT, DistilBERT, XLNet, GPT….” Jan. 2023. [Online]. Available: https://medium.com/mlearning-ai/what-are-the-differences-in-pre-trained-transformer-base-models-like-bert-distilbert-xlnet-gpt-4b3ea30ef3d7

[12] “Financial Opinion Mining and Question Answering.” [Online]. Available: https://sites.google.com/view/fiqa/home

[13] I. W. W. W. C. Committee, “IW3C2.” [Online]. Available: https://www.iw3c2.org/

[14] A. Sinha, S. Kedas, R. Kumar, and P. Malo, “SEntFiN 1.0: Entity-aware sentiment analysis for financial news,” *J Assoc Inf Sci Technol*, vol. 73, Jan. 2022, doi: 10.1002/asi.24634.

[15] avcontentteam, “Scikit-Learn vs TensorFlow: Which One to Choose?” Jan. 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2023/08/scikit-learn-and-tensorflow/

[16] S. Kumar, “Natural Language Processing – Sentiment Analysis using LSTM.” Jan. 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/natural-language-processing-sentiment-analysis-using-lstm/

[17] R. K. Behera, M. Jena, S. K. Rath, and S. Misra, “Co-LSTM: Convolutional LSTM model for sentiment analysis in social big data,” *Inf Process Manag*, vol. 58, p. 102435, Jan. 2021, doi: 10.1016/j.ipm.2020.102435.

[18] C. Voskoglou, “What is the best programming language for Machine Learning?” Jan. 2017. [Online]. Available: https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7

[19] “Why is Python the Best-Suited Programming Language for Machine Learning?” Jan. 2019. [Online]. Available: https://www.geeksforgeeks.org/why-is-python-the-best-suited-programming-language-for-machine-learning/

[20] R. Ozminkowski, “Garbage In, Garbage Out.” Jan. 2021. [Online]. Available: https://towardsdatascience.com/garbage-in-garbage-out-721b5b299bc1

[21] M. Ali, “NLTK Sentiment Analysis Tutorial: Text Mining & Analysis in Python.” Jan. 2023. [Online]. Available: https://www.datacamp.com/tutorial/text-analytics-beginners-nltk

[22] C. Goyal, “Text Vectorization and Word Embedding | Guide to Master NLP (Part 5).” Jan. 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/part-5-step-by-step-guide-to-master-nlp-text-vectorization-approaches/

[23] J. Brownlee, “A Gentle Introduction to the Bag-of-Words Model.” Jan. 2019. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

[24] J. Korstanje, “The F1 score.” Jan. 2021. [Online]. Available: https://towardsdatascience.com/the-f1-score-bec2bbc38aa6

[25] S. K. Agrawal, “Metrics to Evaluate your Classification Model to take the right decisions.” Jan. 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions

[26] Amanatullah, “Fine-Tuning the Model: What, Why, and How.” Jan. 2023. [Online]. Available: https://medium.com/@amanatulla1606/fine-tuning-the-model-what-why-and-how-e7fa52bc8ddf

[27] pintusaini, “Pre-trained Word embedding using Glove in NLP models.” Jan. 2022. [Online]. Available: https://www.geeksforgeeks.org/pre-trained-word-embedding-using-glove-in-nlp-models/

[28] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global Vectors for Word Representation.” Jan. 2014. [Online]. Available: https://nlp.stanford.edu/projects/glove/