Sentiment Analysis of Financial Statements using Recurrent Neural Networks

Trishala Reddy, Pratiksha Aigal, Vidya Sreekumar, Swasha Kumar
The University of Texas at Dallas
{txr220017, ppa20001, vxs220066, sxk220474}@utdallas.edu

***Abstract*—In today's fast-paced financial landscape, staying informed about market sentiment is crucial for making well-informed investment decisions. The proliferation of digital news sources has led to a significant increase in the volume of financial news headlines, presenting both an opportunity and a challenge for investors. To address this challenge, this study leverages Long Short-Term Memory (LSTM) networks, a type of Recurrent Neural Network (RNN), for sentiment analysis of financial news headlines. The research's objective is to develop an LSTM-based RNN model capable of predicting the sentiment (positive or negative) of financial news headlines. A curated dataset specifically designed for training the model is utilized, focusing on news headlines related to finance and stock market activity obtained. The dataset is annotated with sentiment labels, indicating whether each headline conveys a positive or a negative sentiment.**

Keywords—Financial Statement Analysis, Recurrent Neural Networks, PySpark, LSTM, Map Reduce, Sentiment Analysis

I. INTRODUCTION

In recent years, the financial sector has witnessed a surge in the availability and accessibility of digital news sources, contributing to an unprecedented volume of financial news headlines. With market sentiment playing a pivotal role in investment decision-making, the ability to accurately analyze and interpret this vast amount of textual data has become increasingly vital for investors and financial analysts.

RNNs, a class of artificial neural networks, which are designed to handle sequential data, have demonstrated significant success in natural language processing tasks due to their ability to capture temporal dependencies within sequences of data. The backward-looping connections of RNNs, a kind of artificial neural networks, enable them to store data from historical time steps. They are particularly helpful for assessing time-series data, like financial news headlines, because of this feature, which makes them well-suited for modeling sequences. LSTM networks, a specialized variant of RNNs, offer enhanced capabilities for capturing long-term dependencies by incorporating a memory cell that can retain information over extended time periods. LSTM networks are particularly good at identifying subtle patterns and correlations in financial news headlines, which makes it possible to anticipate sentiment dynamics more accurately in the context of financial sentiment research. These attributes make LSTM networks particularly well-suited for sentiment analysis tasks where understanding the context and temporal dynamics of textual data is crucial.

In this study, implementation of an LSTM-based RNN model for sentiment analysis of financial news headlines is presented. Each headline in the dataset is annotated with sentiment labels, enabling supervised learning techniques to train the LSTM-based RNN model. The model is developed using Python, along with PySpark, a powerful framework for distributed computing and processing large-scale datasets. Python and PySpark harness the computational efficiency and scalability required to analyze the vast amounts of textual data inherent in financial news datasets.

II. BACKGROUND WORK

Sentiment analysis, a potent computational tool, is pivotal in discerning the emotional tone underlying textual data, particularly in the financial domain where understanding market sentiment is crucial for making informed decisions. Analysis of Financial sentiments involve the computational scrutiny of subjective information within financial texts, including news articles, reports, and social media content. Its primary objective is to discern and extract opinions that can serve as predictive indicators of market trends, thereby empowering investors and analysts to make well-informed, data-driven decisions.

The advent of machine learning, particularly deep learning, has revolutionized sentiment analysis by furnishing sophisticated models such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. These models excel in capturing the context and dependencies inherent in sequential data, a characteristic ubiquitous in natural language. RNNs constitute a class of neural networks adept at processing sequential data. Particularly advantageous in sentiment analysis, RNNs possess the capability to handle temporal information and retain memory of previous inputs, thus preserving the semantic meaning conveyed by word sequences.

LSTMs, an advanced variant of RNNs, are meticulously crafted to retain information over prolonged durations. Their efficacy lies in circumventing the vanishing gradient problem, rendering them well-suited for tasks such as sentiment classification, where context from distant portions of the sequence is indispensable. However, financial sentiment analysis confronts distinctive challenges, including the comprehension of intricate financial jargon and the nuances of language prevalent in financial news and reports. The pervasive ambiguity and implicit sentiment inherent in financial texts render accurate sentiment prediction an arduous undertaking.

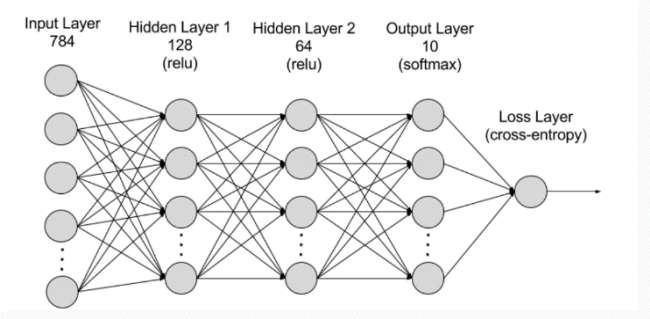
This research endeavors to surmount these challenges by deploying a sophisticated model that integrates RNNs with LSTMs to predict the sentiment of financial texts as positive or negative. By meticulously addressing the subtleties of financial language and harnessing state-of-the-art deep learning techniques, we aspire to augment the accuracy and reliability of financial sentiment analysis.

III. THEORETICAL AND CONCEPTUAL STUDY

A. Recurrent Neural Network

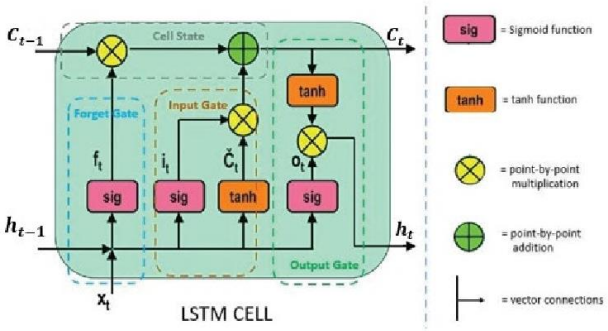
Recurrent neural networks embed recurrent connections within the neural architecture to create a distinct class of models proficient in understanding sequential data. RNNs' inner connections and recurrent connections provide a unique

and outstanding ability to uncover the complexities of time series data because they can maintain knowledge from previous inputs, allowing the model to recognize patterns and trends from one time step to the next. However, traditional RNNs are plagued by the vanishing gradient problem, which severely inhibits the model's ability to learn from sequential data. This is mainly because the gradient that the model uses to learn is gradually degraded over time, making the model less effective at capturing relationships that span over long periods. Long Short-Term Memory networks are a solution to this problem. LSTMs are a variant on RNNs designed to combat the vanishing gradient problem using information flow specifically tailored to this task through several mechanisms, such as gated units and memory cells. Gated units are employed to control the flow of information by choosing which values to remember or forget. LSTM networks represent a significant leap forward in time series data analysis due to their architecture's innovative features and components. LSTMs will change time series data analysis, enabling researchers and experts to discover new patterns and make more accurate predictions based on the data.



B. *Long Short - Term Memory*

Through incorporating the recurrent connections within their architecture, recurrent neural networks exhibit a distinguishing feature, enabling them to learn the underlying structure of sequential data. More specifically, the recurrent connections allow the network access to the preceding inputs and enables it to determine the pattern and trend from one time step to the next. For instance, through recurrent connection, the network can determine the trend of input data and predict the trend of upcoming data. However, traditional RNNs experience a problem that inhibits their ability to learn and capture long-term dependences within sequential data. The vanishing gradient causes the gradient utilized in the training the network to dwindle, thus making it incapable of learning and capturing correlation that occurs over a long period. In this regard, a specific type of RNNs, LSTM networks, was established to solve the vanishing gradient problem. LSTMs comprise memory cells arranged in a series; they can store vital information for a long time. The network includes the input gate, forget gate, and output gate, which regulate the flow of information into and out of the memory cells. The input gate is positioned in between the input unit and the memory cell; it decides which unit is stored in the memory cell. The forget gate determines which memory should be retained, while the output gate decides which memory content is determined.



LSTM networks are used to capture sequences of text for the analysis of financial text, including financial reports, news articles, and market sentiment, among others. The LSTM network accepts input in the form of a sequence of textual inputs. It may also take other relevant information sources related to the inputs like sentiment analysis and key financial indicators. It generates output predictions, such as sentiment class labels or trend forecasts, from the sequence of textual inputs. Additionally, it captures the hidden factors about the input, which include patterns and connections in financial text. Numerous developments are still underway to improve the efficiency, interpretability, and more efficient and adaptable LSTM networks to variable market conditions. Theoretical and conceptual research has been instrumental in contributing to the understanding and knowledge of LSTM networks for financial text analysis. It acts as a source of ideas and research areas for optimal model measures to obtain insights for decision-making in finance.

C. *Understanding use of PySpark / Map Reduce*

One of the primary benefits of PySpark/MapReduce is that it allows you to process vast amounts of data quickly, especially useful when working with advanced machine learning models such as the LSTM-RNN used to predict financial trends. The Hadoop framework consists of a MapReduce implementation and a distributed file system named HDFS [3]. PySpark is a Python application programming interface that makes it easier to divide up data processing tasks among multiple nodes in a cluster. Because of the millions of records available in financial accounts, this capability is vital. Additionally, because the models are excellent at handling sequence data, LSTM-RNN makes some of their best forecasting using historical financial data. This is since time-series data, which is typical in financial markets, has complicated patterns and temporal connections that benefit from these models. Moreover, this relationship structure is highly enhanced by the level at which the PySpark/MapReduce program processes data. Because even the most difficult computational tasks can be completed quickly due to the framework's ability to distribute processing responsibilities across a network of systems. Furthermore, new data, which gains proficiency through PySpark's real-time data processing capability, allows machines to learn and improve their forecasting capabilities. In summary, pairing PySpark/MapReduce and LSTM-RNN is reliable for financial trend prediction. Not only does the technology increase the scale and real-time nature of the financial analysis models, but it also accelerates data processing and model training. Thus, financial and investment professionals can make more accurate predictions and informed decisions.

D. Root Mean Squared Error (RMSE)

RMSE gives a comprehensive view of the overall performance of the model across the entire dataset. A lower RMSE indicates that the model's predictions tend to be closer to the actual values, whereas a higher RMSE suggests greater deviation between predicted and actual values, indicating potential areas for improvement in the model.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

IV. PERFORMANCE MEASURE

Several parameters are used to evaluate the model's performance. Using the model's ratings on the precision, accuracy, recall this study assessed it.

The following equations can be used to determine the values of accuracy, precision, recall:

$$Accuracy\ (\%) = \frac{(TP+TN)}{(TP+FP+FN+TN)} \times 100$$

$$Precision\ (\%) = \frac{(TP)}{(TP+FP)} \times 100$$

$$Recall\ (\%) = \frac{(TP)}{(TP+FN)} \times 100$$

V. METHODOLOGY

A. Dataset Preprocessing

The dataset was intentionally selected to train a machine learning model that could investigate the sentiments regarding the news headlines, and the potential impact of the headlines on many of the companies' stock prices. Analyzing the sentiments as positive or negative based on the anticipated impact on the stock price is the main goal of the model trained on this dataset.

Using the Natural Language Toolkit (nltk) and Python's re and string libraries, a number of transformations are applied to the textual material to prepare it for sentiment analysis. To maintain consistency, text inputs from financial headlines are first transformed to lowercase. The next steps are to use regular expressions to remove text enclosed in square brackets, remove punctuation, and exclude terms that contain numbers. This is further improved by utilizing nltk to remove often occurring stop words so that the text's more powerful words can be highlighted.

After being cleaned, PySpark a potent library with distributed computing capabilities for managing big datasets, is used to parse the text. PySpark's Tokenizer divides the cleaned text into distinct terms. Each token is given a distinct numerical index that is kept in a dictionary to support neural network modeling. Sequences of tokens are then padded to a uniform length established by the longest sequence in the dataset. For batch processing in deep learning models, this constant sequence length is essential since it guarantees that the neural network's input layer gets consistently formatted data. By utilizing PySpark for scalable and effective data processing, these procedures together ready the dataset for efficient sentiment analysis model training and testing.

B. Implementation

a) Model Selection and Training: The model considered for the work is a hybrid RNN-LSTM. This was chosen as such a model has the potential for implementing sequential data. In this work, a sequential dataset is analyzed as the text documents need to be represented as words. Therefore, ensuring the sequence of data when analyzing financial text, where the actual message may last several sentences.

b) The LSTM was specifically taken to avoid the vanishing gradient problem common in traditional RNNs. The model was put through training on the training dataset using the sigmoid and tanh activation functions and optimization of the model using hyperparameter tuning. Evaluation metrics such as accuracy and precision were used to assess the model and ensure the model is reliable.

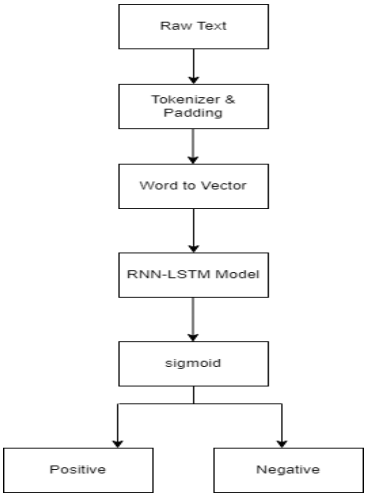


Fig: Implementation of RNN-LSTM model

VI. RESULT AND ANALYSIS

a) Hyperparameter tuning:

Just for the purpose of hyperparameter tuning a Keras LSTM model was used to understand the best fitting parameters for our data. The best found hyperparameters were then used to build the LSTM model from scratch.

Below is the log of few of the experiments conducted:

N/o	Hyperparameters	Results
1	units=32, learning_rate=0.01, epochs=5	Train Accuracy:0.9752066115702479, Test Accuracy: 0.8172588832487731 Train RMSE: 0.1574591643244434, Test RMSE: 0.42748229992745784 Train Precision: 0.9862511457378552 Test Precision: 0.8304498269896193
2	units=32, learning_rate=0.01, epochs=10	Train Accuracy: 0.9841068022886205, Test Accuracy: 0.8350253807106599 Train RMSE: 0.1260682264148248, Test RMSE: 0.40617067753512204

		Train Precision: 0.9944802207911684, Test Precision: 0.8639705882352942
3	units=32, learning_rate=0.01, epochs=15	Train Accuracy: 0.9910998092816274, Test Accuracy: 0.8096446700507615 Train RMSE: 0.09434082212050379, Test RMSE: 0.43629729537236256 Train Precision: 0.994535519125683, Test Precision: 0.8405797101449275
4	units=32, learning_rate=0.001, epochs=5	Train Accuracy: 0.9796567069294342, Test Accuracy: 0.8197969543147208 Train RMSE: 0.14262991646413384, Test RMSE: 0.42450329290275146 Train Precision: 0.9881170018281535, Test Precision: 0.8287671232876712
5	units=32, learning_rate=0.001, epochs=10	Train Accuracy: 0.9910998092816274, Test Accuracy: 0.8197969543147208 Train RMSE: 0.09434082212050379, Test RMSE: 0.42450329290275146 Train Precision: 0.990072202166065, Test Precision: 0.8333333333333334
6	units=32, learning_rate=0.001, epochs=15	Train Accuracy: 0.993006993006993, Test Accuracy: 0.8223350253807107 Train RMSE: 0.08362420100070908, Test RMSE: 0.42150323203896 Train Precision: 0.9945504087193461, Test Precision: 0.8458781362007168
7	units=32, learning_rate=1e-05, epochs=5	Train Accuracy: 0.9726636999364272, Test Accuracy: 0.8299492385786802 Train RMSE: 0.16533692891659985, Test RMSE: 0.4123721152324922 Train Precision: 0.9907149489322191, Test Precision: 0.855072463768116
8	units=32, learning_rate=1e-05, epochs=10	Train Accuracy: 0.9917355371900827, Test Accuracy: 0.8274111675126904 Train RMSE: 0.09090909090909091, Test RMSE: 0.4154381211291396 Train Precision: 0.9918552036199095, Test Precision: 0.8469750889679716
9	units=32, learning_rate=1e-05, epochs=15	Train Accuracy: 0.9923712650985378, Test Accuracy: 0.817258883248731 Train RMSE: 0.08734262934822934, Test RMSE: 0.42748229992745784 Train Precision: 0.9936479128856625, Test Precision: 0.8374558303886925
10	units=64, learning_rate=0.01, epochs=5	Train Accuracy: 0.9758423394787031, Test Accuracy: 0.8248730964467005 Train RMSE: 0.15542734804820188, Test RMSE: 0.41848166453657143 Train Precision: 0.9715808170515098, Test Precision: 0.8211920529801324
11	units=64, learning_rate=0.01, epochs=10	Train Accuracy: 0.9898283534647171, Test Accuracy: 0.8350253807106599 Train RMSE: 0.10085458113185984, Test RMSE: 0.40617067753512204 Train Precision: 0.9918330308529946, Test Precision: 0.8613138686131386
12	units=64, learning_rate=0.01, epochs=15	Train Accuracy: 0.9917355371900827, Test Accuracy: 0.8274111675126904 Train RMSE: 0.09090909090909091, Test RMSE: 0.4154381211291396 Train Precision: 0.9900811541929666, Test Precision: 0.8373702422145328
13	units=64, learning_rate=0.001, epochs=5	Train Accuracy: 0.9764780673871583, Test Accuracy: 0.8197969543147208 Train RMSE: 0.1533686167794497, Test RMSE: 0.42450329290275146 Train Precision: 0.9898617511520738, Test Precision: 0.8356643356643356
14	units=64, learning_rate=0.001, epochs=10	Train Accuracy: 0.9866497139224412, Test Accuracy: 0.8096446700507615 Train RMSE: 0.1155434380549532, Test RMSE: 0.43629729537236256 Train Precision: 0.99179580674567, Test Precision: 0.8333333333333334

From the experiment the below hyperparameters gave the best results with an accuracy of 83%.
Best hyperparameters: {'units': 32, 'learning_rate': 0.01, 'epochs': 10}

b) Evaluating the model using loss function

The trained model was evaluated using root mean squared error (RMSE)

c) Activation Function

The sigmoid activation function was used for training the LSTM - RNN Model as we are trying to solve binary classification problem.

Cell	Activation Function
Input Cell	Sigmoid
Output Cell	Sigmoid
Forget Cell	Sigmoid
Candidate Cell	Tanh

d) Number of epochs

Based on the hyper-parameter tuning the model was run for 10 epochs.

e) Accuracy and Running time.

Model	Train Accuracy	Test Accuracy	Run time
LSTM without Map Reduce	0.692	0.691	111.91 seconds (about 2 minutes)
LSTM with Map Reduce	0.418	0.407	96.40seconds (about 1 and a half minutes)

Notice how running LSTM with Map Reduce helps reduce the runtime when compared to the one without. But however, we have a serious decrease in the accuracy of the model. One possible reason could be that LSTM is a sequential model, meaning that it builds on top of the previously learned weights.

In the map reduce implementation we tried to parallelize the weight computation in each epoch and added them up in the reduce phase. But as seen this affects the learning capabilities of the model. For implementing a full MapReduce based LSTM model would require core architectural changes in the LSTM model, which leaves scope for future work.

Sentiment analysis of financial statements using RNNs has demonstrated the effectiveness of deep learning techniques in extracting valuable insights from textual data within the financial domain. By implementing an LSTM RNN model from scratch and fine-tuning its hyperparameters using Keras LSTM models, we achieved robust performance in capturing sentiment trends with good accuracy. These findings underscore the potential of recurrent neural networks to discern subtle nuances in financial texts, thereby enabling informed decision-making and risk management strategies in financial markets. Moving forward, continued research and development in this field offers promising opportunities for leveraging artificial intelligence to enhance financial analysis and forecasting capabilities.

VII. Conclusion and Future Work

The research demonstrated the application of a hybrid RNN-LSTM model, used to predict the sentiment of financial texts. Despite the difficulties arising from linguistic subtleties and data preprocessing, the model demonstrated impressive performance metrics. In the future, more varied datasets including those from various financial subdomains can be incorporated into the model to improve its accuracy. The model's comprehension of intricate financial storylines may also be enhanced by investigating the merging of transformer

models and attention mechanisms. Increasing the number of layers in the model would also aid in increasing its accuracy and precision.

REFERENCES

[1] N. M. Rezk, M. Purnaprajna, T. Nordström and Z. Ul-Abdin, "Recurrent Neural Networks: An Embedded Computing Perspective," in IEEE Access, vol. 8, pp. 57967-57996, 2020, doi: 10.1109/ACCESS.2020.2982416.

[2] B. Nath Saha and A. Senapati, "Long Short Term Memory (LSTM) based Deep Learning for Sentiment Analysis of English and Spanish Data," 2020 International Conference on Computational Performance Evaluation (ComPE), Shillong, India, 2020, pp. 442-446, doi: 10.1109/ComPE49325.2020.9200054.

[3] Borthakur D. HDFS architecture guide[J]. Hadoop Apache Project. http://hadoop. apache. org/common/docs/current/hdfs-design. pdf, 2008.

[4] X. Man, T. Luo and J. Lin, "Financial Sentiment Analysis(FSA): A Survey", 2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS), pp. 617-622, May 2019.

[5] K. L. Kohsasih, B. H. Hayadi, Robet, C. Juliandy, O. Pribadi and Andi, "Sentiment Analysis for Financial News Using RNN-LSTM Network," 2022 4th International Conference on Cybernetics and Intelligent System (ICORIS), Prapat, Indonesia, 2022, pp. 1-6, doi: 10.1109/ICORIS56080.2022.10031595.