Financial News Sentiment Analysis Project Documentation

About the Project

The Financial News Sentiment Analysis project aims to create a machine learning tool that analyzes the sentiment of financial news articles, classifying them as Negative, Neutral, or Positive. The goal is to achieve over 95% accuracy using a dataset of about 5,000 rows. This tool is helpful for investors and analysts who need quick insights into market trends to make better decisions. The project uses a Streamlit web app for easy access, where users can input news text and get instant results. It relies on Python, TensorFlow for the model, and NLTK/NLPAug for data handling. The project addresses the need for automated sentiment analysis to save time and reduce bias in financial decision-making.

The motivation comes from the growing demand for fast, reliable sentiment analysis in finance, where manual reviews are slow and inconsistent. The development involved testing and refining the model to handle data challenges, ensuring it works well for real-world use. This tool combines advanced AI with practical application, making it useful for tracking market sentiment. The documentation explains the entire process, from data preparation to deployment, helping users understand how to use and benefit from the project in various financial contexts.

Preprocessing: Creating Three Datasets

The project starts with an original dataset of about 5,000 rows of financial news text with labels (-1, 0, 1). Initial cleaning removed rows with missing or invalid text, reducing it to 4,843 rows. This step ensured the data was clean and reliable, which is vital for accurate sentiment analysis in finance where data quality affects results. The cleaning process laid a strong foundation for further improvements by eliminating noise that could confuse the model.

Next, an augmented dataset was created by doubling the size to around 9,686 rows using NLPAug with synonym replacement at a 30% rate. This added variety to the data, helping the model learn from more examples and improve its ability to recognize different sentiment expressions. Augmentation was necessary because the original dataset was limited, and financial news often uses varied language that needs broader exposure for effective training.

Finally, a balanced dataset was made by oversampling to match the neutral class (5,752 rows) with negative (1,208 to 5,752) and positive (2,726 to 5,752) classes, resulting in about 17,256 rows. This balancing prevented bias toward the neutral class, which was common in the data. The process of cleaning, augmenting, and balancing was crucial to ensure the model could learn equally from all sentiments, making it suitable for the nuanced language of financial news.

Models Used

The project uses a hybrid Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) model built with TensorFlow. This combination is ideal for text analysis, as CNNs extract key features and LSTMs understand the sequence of words, which is important in financial news where context matters. The model was designed to handle the complexities of this domain, balancing performance and efficiency through careful layer planning. This approach ensures the model can process sequential data effectively.

The model starts with an Embedding layer that converts text into vectors using a 30,000-word vocabulary and 100-dimensional output. A Conv1D layer with 256 filters and a 5-unit kernel uses ReLU to find local patterns, followed by MaxPooling1D with a 5-unit pool to reduce size. A Bidirectional LSTM with 128 units captures context from both directions, aided by BatchNormalization and 0.4 Dropout to avoid overfitting. Another Bidirectional LSTM with 128 units deepens the analysis, ending with a Dense layer of 3 units using softmax for classification.

Training used categorical cross-entropy loss and the Adam optimizer, running up to 50 epochs with a 64-batch size. Early stopping with a 5-epoch patience on validation accuracy saved the best model. The trained model and tokenizer are saved as 'sentiment_model.h5' and 'tokenizer.pkl' for use in the app. This structured design reflects a thoughtful approach to building a robust sentiment analysis tool.

Results

Training completed 108 steps in about 6 seconds each, showing good efficiency with the GPU in Colab. This indicates the model handled the 17,256-row dataset well, proving the setup was optimized for performance despite the initial warning, which is a common TensorFlow note and not a critical issue.

The model was tested on a 20% set of 3,452 samples, achieving an accuracy of 0.9652, meaning 96.52% of predictions were correct. Precision, recall, and F1-score were all around 0.965, showing balanced performance across metrics. This high consistency suggests the model generalizes well, thanks to the balanced dataset, with no major overfitting or underfitting. The results meet the goal of over 95% accuracy, validating the training approach.

The classification report shows strong class performance: Negative (1,158 samples) had 0.97 precision and 1.00 recall with a 0.98 F1-score, excelling at detection. Neutral (1,171 samples) had 0.97 precision, 0.94 recall, and 0.95 F1-score, with a slight recall dip. Positive (1,123 samples) had 0.96 for all metrics, showing consistency. Macro and weighted averages of 0.97 confirm the model's reliability across all sentiments, highlighting the success of the preprocessing and model design.

Usage Instructions

To start, install the required libraries by creating a 'requirements.txt' file with pandas, numpy, scikit-learn, tensorflow, streamlit, and nlpaug, then run 'pip install -r requirements.txt' in a terminal. This step ensures all tools are ready, whether you're working locally or in Colab, and is simple for anyone with basic Python skills. It sets up the environment for both training and running the app, making the project accessible.

Launch the Streamlit app by navigating to the project folder in the terminal and running 'streamlit run app.py'. This starts the server, usually opening the app at 'http://localhost:8501' in your browser. If it doesn't open, type the URL manually. The app offers a text area for inputting news, and clicking "Analyze" shows the sentiment and scores, designed for ease of use without technical expertise.

Use the app by entering full news sentences or paragraphs for accurate results. It will display the sentiment (-1, 0, or 1) with probability details. Keep the model and tokenizer files in their 'model' and 'data' folders to ensure proper functioning. This setup provides a practical way to test and deploy the tool for financial analysis, suitable for real-time use.

Future Improvements

One improvement could be reducing the neutral bias in app predictions by adding more varied negative examples to the training data or tweaking the prediction thresholds. This would make the model better at catching subtle negative hints in news, potentially pushing accuracy higher than 97%. Collecting new data or adjusting thresholds could be tested to refine performance, offering a way to handle edge cases more effectively.

Another enhancement is adding financial indicators like stock prices or economic metrics to the model. This would give extra context to improve analysis of complex financial stories. It would need an updated dataset with these features and a modified model, which could be a big but beneficial step. This would make the tool more versatile for detailed financial insights.

Considering advanced methods like transfer learning with BERT could also boost the model. This would use pre-trained knowledge to better handle financial jargon and nuances, though it requires more computing power and tuning. Balanced against the current model's success, these improvements would keep the project relevant as financial data needs grow.

Ankit Ambasta
ambastaankit8@gmail.com