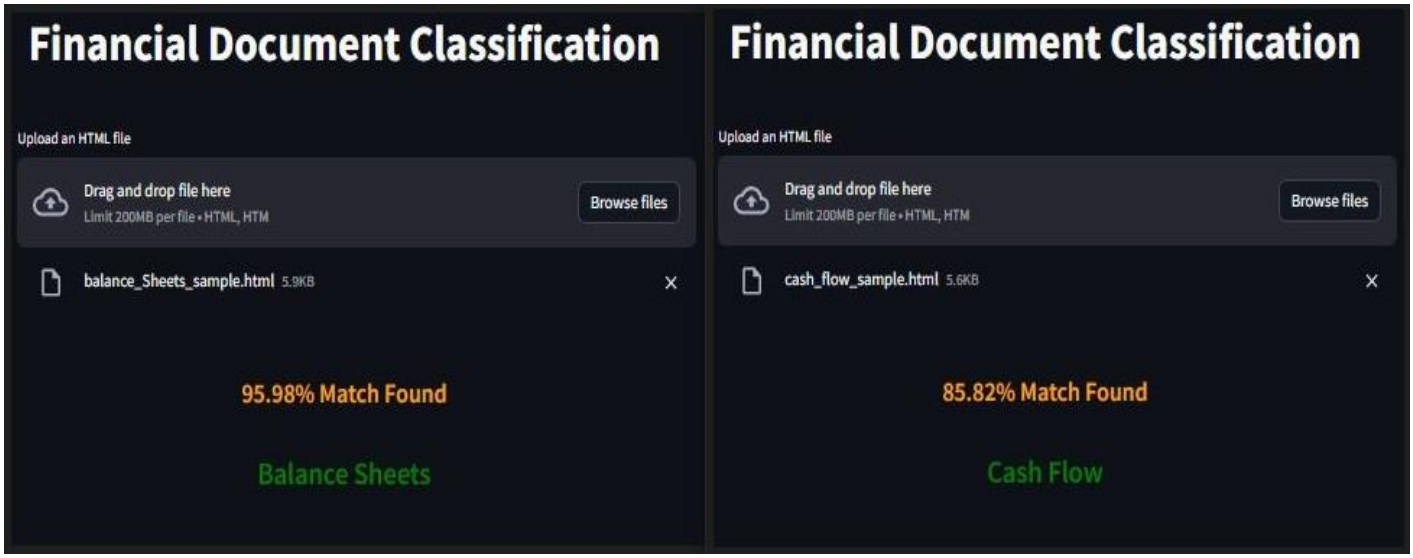
Financial Document Classification using Deep Learning

Introduction

Managing and classifying financial documents such as Balance Sheets, Cash Flow statements, Income Statements, Notes, and Other Documents manually is time-consuming and prone to errors. This project addresses these challenges by automating the classification process using deep learning techniques. Leveraging TensorFlow and a Bidirectional Long Short-Term Memory (LSTM) RNN, we accurately categorize these documents. The model is integrated into a user-friendly Streamlit application and deployed on the Hugging Face platform, ensuring high accuracy and efficiency in financial document management.



Application: https://huggingface.co/spaces/gopiashokan/Financial-Document-Classification

Environment Setup

Ensure the following libraries are installed:

```
pip install tensorflow
pip install spacy
pip install nltk
pip install gensim
pip install imblearn
pip install numpy
pip install pandas
pip install streamlit
pip install streamlit_extras
pip install beautifulsoup4
pip install matplotlib
pip install wordcloud
pip install scipy==1.12
pip install “https://github.com/explosion/spacy-models/releases/download/en_core_web_lg-3.7.1/en_core_web_lg-3.7.1-py3-none-any.whl”
```

Data Collection

The dataset comprises HTML files organized into five distinct folders, namely Balance Sheets, Cash Flow, Income Statement, Notes, and Others. These folders represent various financial document categories. You can access the dataset via the following download link.

Download Link: https://drive.google.com/file/d/1yj_ucy-VuX7fjKAQsR23ViTc-odb-eD-/view

Data Preprocessing

- **Text Processing**: BeautifulSoup was utilized to extract content from HTML files, and NLTK was employed for word tokenization to split the text into smaller tokens. Furthermore, Spacy was used to convert to the base root form in the lemmatization process, and to remove stop words, special characters, and duplicates in each sentence.

- **Word Embedding**: The Word2Vec model was trained using the tokenized sentences list with a vector size of 300 dimensions. In the embedding process, all text was converted into a 300-dimensional vector, and the target classes were encoded accordingly. Subsequently, the Word2Vec model was saved for inference.

- **Imbalance Dataset handling**: Implemented the SMOTETomek oversampling technique to address the imbalance in the dataset, ensuring each class is represented adequately. This involved generating synthetic samples to balance the dataset for each class, thereby improving the model's ability to learn from all classes equally. The SMOTETomek approach combines the SMOTE (Synthetic Minority Over-sampling Technique) and Tomek links to create a balanced distribution of samples, enhancing the robustness and reliability of the model.

- **Data Preparation:** Features and target variables were transformed into tensors using TensorFlow, facilitating compatibility with deep learning models. Subsequently, a TensorFlow dataset was constructed with a batch size of 32 to ensure efficient model training.

- **Data Splitting:** The dataset was divided into training (80%), validation (10%), and testing (10%) sets using a custom function. This partitioning strategy ensured an appropriate distribution of data for model training, validation, and evaluation, thereby enhancing the robustness of the trained model.

Model Development

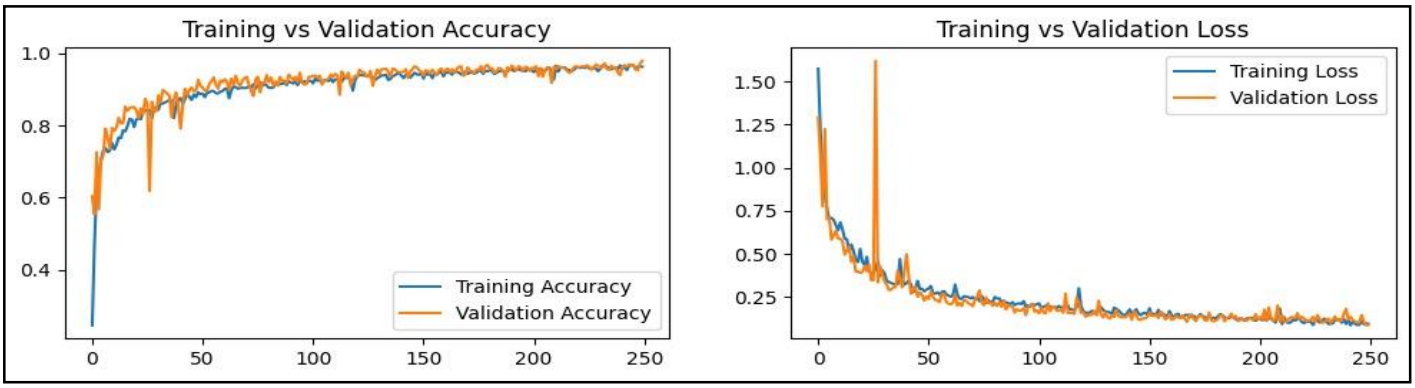
- **Optimized Data Pipeline:** An optimized data pipeline was constructed to improve the performance of the model during training. This pipeline incorporated cache, shuffle, and prefetch functions, enabling seamless and efficient data processing, thereby reducing training time and resource consumption.

- **Model Architecture:** A Recurrent Neural Network (RNN) architecture, specifically Bidirectional Long Short-Term Memory (LSTM), was developed for model training. This architecture consisted of multiple bidirectional LSTM layers supplemented with a dropout layer, effectively mitigating over-fitting issues by randomly dropping neurons during the training process.

- **Activation Functions:** The model utilized tanh activation as the default activation function, ensuring non-linearity in the model's outputs. Additionally, sigmoid activation was employed for the forget gate within the LSTM layers, enhancing the model's ability to retain or forget information over time. The multiclass classification output layer employed softmax activation, facilitating the probabilistic interpretation of model's predictions across multiple classes.

- **Optimization and Loss Function:** During model training, the Adam optimizer was utilized to minimize the SparseCategoricalCrossentropy loss function. This combination effectively optimized the model's parameters, enabling efficient convergence during the training process.

- **Model Evaluation:** The developed model achieved commendable performance metrics, demonstrating its efficacy in classifying financial documents. The training accuracy of 0.96, validation accuracy of 0.98, and testing accuracy of 0.99 underscore the robustness and generalization capability of the trained model.



Model Deployment

- **Model Saving:** Upon successful training, the trained model was saved to facilitate inference on new HTML documents. This step ensured that the model parameters were preserved, enabling seamless integration into the deployment environment.

GitHub: https://github.com/gopiashokan/Financial-Document-Classification-using-Deep-Learning.git

- **Application Development:** A user-friendly Streamlit application was developed to allow users to upload new HTML documents for classification. The application provided a simple interface for users to interact with, displaying the predicted class and associated confidence scores. Additionally, the application showcased the uploaded document, enhancing the interpretability of the classification results.

- **Deployment on Hugging Face:** The Streamlit application was deployed on the Hugging Face platform, enabling easy access for users to utilize the model for document classification. By deploying on Hugging Face, users can seamlessly upload new HTML documents, receive classification predictions, and visualize the document, enhancing the overall user experience.

Conclusion

In conclusion, this project successfully addressed the challenge of classifying financial documents using deep learning techniques. By employing methods such as data preprocessing, word embedding, data balancing, and model development, we achieved high accuracy in classifying financial documents. The developed Streamlit application provides a user-friendly interface for document classification, enhancing accessibility and usability.

References

- SpaCy Documentation: https://spacy.io/usage
- NLTK Documentation: https://www.nltk.org/
- TensorFlow Documentation: https://www.tensorflow.org/
- Gensim Documentation: https://radimrehurek.com/gensim/
- Streamlit Documentation: https://docs.streamlit.io/