MF815 Advanced Machine Learning Applications for Finance - Final Project

Chi-Lin Li

1 Introduction

This project explores advanced machine learning applications for finance, emphasizing the use of pre-trained models and their impact on predictive performance. The study employs various machine learning techniques and natural language processing (NLP) tools to analyze financial data, focusing on text-based feature engineering and embedding methods like Word2Vec, Sentence BERT, and RoBERTa. The objective is to enhance data understanding and model accuracy, leveraging techniques such as TF-IDF vectorization, cosine similarity, and advanced graph-based visualization to uncover hidden insights in financial texts. By implementing algorithms like Random Forest, LightGBM, and XGBoost, the project aims to highlight the significance of optimization and model selection in improving precision, recall, and F1-score metrics. Furthermore, the project underlines the importance of semantic relationships and feature engineering in developing robust models for finance-related tasks.

2 Methodologies

2.1 Preprocessing

The dataset provides insights into the characteristics of various financial funds, revealing gaps in data and trends in investment strategies, leverage use, and portfolio composition. A noticeable portion of the entries (447) lacks performance fee information, indicating data collection gaps. The predominant investment strategies are low-risk, with 'Equity Long Only' and 'Fixed Income Long Only' being the most prevalent. Notably, 192 funds leverage borrowed money to enhance returns. Asset quality primarily includes investment-grade securities and listed equities, but some funds also invest in higher-risk sub-investment-grade securities. Most funds have diversified portfolios, though a smaller number (92) prefer concentrated investments, increasing their risk and potential returns. The dataset is split for model training and evaluation, first dividing into training (70%) and temporary (30%) sets, then further splitting the temporary set into validation (20% of the total) and test sets (10% of the total). This structured approach ensures the data is appropriately allocated for model training, validation, and testing, with reproducible results ensured by setting 'random_state=42'.

2.2 Skip Gram Model

The initial step involves cleaning and tokenizing textual data to prepare it for machine learning. The tokenizer function normalizes the text by removing newline and tab characters, converting it to lowercase, and then uses word tokenization to split it into individual words. English stopwords are filtered out, and non-alphabetic characters are removed to refine the text to significant words. This preprocessed text data is then used to train a Word2Vec model with specific parameters. The model processes the data in batches of 128 and iterates over the dataset twice (2 epochs). The embedding vectors are set to a dimensionality of 50, with a vocabulary size capped at 5,000 words, excluding those that appear fewer than 10 times, focusing the model on the most prevalent vocabulary.

- **Batch Size**: The number of samples processed at once is set to 128.
- **Embedding Dimensionality**: The dimensionality of word embeddings is 50.
- **Vocabulary Size**: The maximum number of unique words is 5000.
- **Context Window**: Predicting words within a window of 3 words on each side of the target word.
- **Epochs**: The number of iterations over the dataset is set to 2.
- **Rare Words Exclusion**: Words appearing fewer than 10 times are excluded.

Furthermore, a dictionary is constructed to map words to unique integers, replacing rare words with a universal **UNK** (unknown) token to handle out-of-vocabulary words efficiently during training. This setup aims to optimize both the learning process and the quality of the word vectors produced.

2.2.1 A Word Embedding Dictionary

The batchgenerator function is designed to continuously generate training batches and corresponding labels for the skip-gram model. It operates within an infinite loop, creating batches where each batch consists of words and their respective context words as labels, encoded via the previously defined one-hot function. The function ensures that each batch is balanced with an equal number of words and context pairs, facilitated by parameters such as batchsize, numskips, and skipwindow. The loop intelligently manages the data indices, using a buffer to store the current sliding window of words to handle dependencies within the data stream efficiently. As the training progresses, it outputs batches of **one-hot encoded word vectors** and their respective **context vectors**, ready for processing by the neural network model. This setup is crucial for effective learning in models that predict the context given a word, as in the skip-gram architecture.

The skip-gram model implemented in this project utilizes an **autoencoder** architecture to learn dense word embeddings. The model consists of a symmetric autoencoder with an input layer that matches the vocabulary size. The encoding layer then compresses this input into a dense representation of the desired embedding size using a linear activation function. The decoded layer reconstructs the original vocabulary distribution using a **softmax** activation function. The model is compiled with the **Adam** optimizer, known for its adaptive learning rate capabilities, and binary cross-entropy loss, which helps the model converge effectively. This architecture is designed to efficiently train and produce high-quality word embeddings that capture semantic relationships within the data.

2.2.2 Training and Vectorization

The model is trained using a **generator-based** approach to efficiently handle large datasets, allowing it to process batches in a streaming manner. The generator feeds batches of data into the model, utilizing a specified batch size and training for a set number of epochs. The generator's design ensures that the model receives training samples with a balanced distribution of word pairs for effective learning.

After training, the encoder component of the autoencoder model is used to generate word vectors. This process involves converting words into one-hot encoded vectors and then passing them through the encoder to produce dense, meaningful word representations. These vectors map words into a lower-dimensional, continuous vector space, making them suitable for various natural language processing (NLP) tasks. This transformation aids in capturing **semantic relationships** between words, enabling advanced text analysis and manipulation.

2.3 Build Knowledge Base

A focused approach is taken to utilize word embeddings for identifying semantically related terms within a specific domain, such as investment and financial management. The script defines a list of key terms that are central to understanding various aspects of investment strategies, including operational fees, portfolio management, market risks, and financial instruments like derivatives and equities. This list forms the basis for generating a knowledge base that maps each keyword to other terms closely related in semantic space.

The core functionality is encapsulated in the 'CustomWord2Vec' class, which is designed to work with pre-loaded word vectors—likely trained on financial texts to capture domain-specific nuances. The class includes a method 'mostsimilar' that calculates the cosine similarity between the vector representation of a given keyword and all other vectors in the model. This similarity metric quantifies how closely words are related based on their vector orientations in multi-dimensional space, enabling the identification of the top five words most similar to each keyword. This method is robust in drawing out nuanced relationships, such as between "Interest" and "Rate" or "Derivatives" and "Risk," which are crucial for informed financial decision-making. The resulting 'knowledge_base' is a dictionary where each entry maps a keyword to a list of its most contextually similar words, providing a valuable resource for automated systems in finance to understand and process information more effectively.

2.3.1 Directed Graph

Using the NetworkX library, a directed graph (DiGraph) is constructed to visually represent the relationships between keywords and their related terms based on semantic similarity. This process starts with importing the NetworkX module and creating an empty directed graph ‘G‘ to hold the nodes and edges. For each keyword in the ‘relatedtermsdict‘, nodes are added for the keyword itself and its related terms, while edges are drawn between them to reflect the strength of their association through weights derived from cosine similarity. This method highlights the semantic relationships between the terms, visually emphasizing clusters and connections in the dataset using a spring layout. The graphical representation provides intuitive insights into the structure of the data, allowing further analysis or visualization of keyword associations.

2.4 Distance Measurement

To calculate the semantic similarity between selected summaries and knowledge base entries, TF-IDF (Term Frequency-Inverse Document Frequency) and cosine similarity measures are used. First, all texts are combined to maintain a consistent vocabulary, and a ‘TfidfVectorizer‘ converts the text data into a TF-IDF matrix, where rows represent documents and columns contain unique word TF-IDF scores. This yields two TF-IDF submatrices: one for summaries and one for knowledge base entries. The cosine similarity, calculated as similarity $= \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$, measures the cosine of the angle between two TF-IDF vectors, ranging from 1 (identical) to 0 (orthogonal). This allows the calculation of a similarity matrix that quantifies the relationship between each summary and knowledge base entry, which is vital for information retrieval, document classification, and content recommendation, enhancing the performance of these tasks by understanding text relationships.

2.5 Classification Algorithm

Light GBM (Light Gradient Boosting Machine), XGBoost (eXtreme Gradient Boosting), and Random Forest are popular and powerful machine learning algorithms that handle large-scale data efficiently. Light GBM is notable for its speed and efficiency, achieved through gradient-based one-sided sampling and exclusive feature bundling. It improves on the gradient boosting decision tree algorithm by using histogram-based learning, which reduces memory usage and speeds up gradient computations. The updates for model parameters in Light GBM follow $\Delta\theta = -\eta \cdot \frac{\partial L}{\partial \theta}$, where η represents the learning rate and L is the loss function. XGBoost, known for its flexibility and performance, extends gradient boosting by handling missing values, implementing regularization, and tree pruning. Its regularization term, $\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2$, helps control the model's complexity, making it robust against overfitting. Random Forest, a popular ensemble learning method, builds multiple decision trees during training and outputs the mode of their predictions. Its use of bagging and feature randomness provides robust results against overfitting and high variance. All three algorithms offer comprehensive options for tuning and are adaptable to various data science tasks and challenges.

2.6 Encoding

To encode the test data for machine learning models, various transformations and feature engineering steps are used to handle categorical variables and extract text-based features. Initially, irrelevant columns like ”id,” ”fundname,” and ”Performance fee?” are removed, while ’Leverage?’ is encoded using a ‘LabelEncoder‘ to convert it into a numeric format. One-hot encoding with ‘pd.get_dummies‘ is applied to ’Portfolio composition’ and ’Concentration’ to avoid multicollinearity by using ‘dropfirst=True‘. For text-based insights, TF-IDF vectorization is performed on summaries and a predefined knowledge base, converting text into a numeric form that emphasizes the importance of words. Cosine similarity is calculated between the TF-IDF vectors to generate a matrix capturing textual relationships, from which maximum and mean similarity scores are extracted. These features, combined with numerical data, form a comprehensive dataset for predictive modeling. Meanwhile, the ’Investment Strategy’ target variable is encoded using numeric identifiers to facilitate machine learning.

2.7 Sentence BERT & RoBERTa

In the BERT embedding, I adopt two state-of-the-art transformer models, Sentence BERT and RoBERTa (Robustly Optimized BERT Approach), known for their powerful natural language processing capabilities. BERT, loaded with its base, uncased variant, processes text bidirectionally to understand context effectively in both directions. Meanwhile, RoBERTa, an optimized and improved variant of BERT, is loaded in its larger configuration, enhancing its ability to comprehend language through optimizations like removing the next sentence prediction objective and utilizing larger training batches. Both models excel at extracting rich semantic information from text, making them suitable for a wide range of NLP tasks, including text classification and sentiment analysis.

3 Empirical Results

3.1 Knowledge Base

In the Figure 1, nodes represent various financial terms, while the edges connecting them represent relationships or similarities between these terms. Here’s a explanation for the terms and relationships shown:

- **Clusters of Related Terms**: The nodes are arranged in a circular layout, and terms that are closely related to each other tend to be near each other.

- **Term Types**: Terms can be broadly classified into several financial categories
  - Investment and Returns: Portfolio, Returns, Asset, and Equity.
  - Management and Strategies: Performance, Management, Dynamic, and Strategies.
  - Expenses and Fees: Fees, Expense, Annual, and Total.
  - Market and Trading: Market, Credit, Short, and Derivatives.
  - Risk and Asset Classes: Risk, Leverage, Debt, Emerging, and Non-U.S.

The proximity of nodes and the thickness of the edges could represent the strength of the relationships between terms. Closer and thicker edges might indicate a stronger relationship or similarity, while thinner or more distant connections suggest weaker relationships.

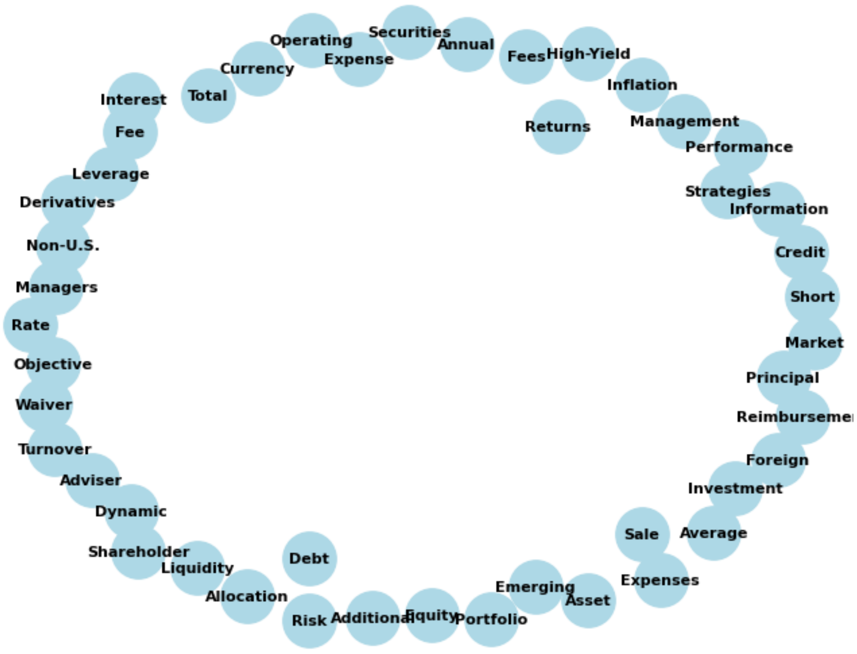
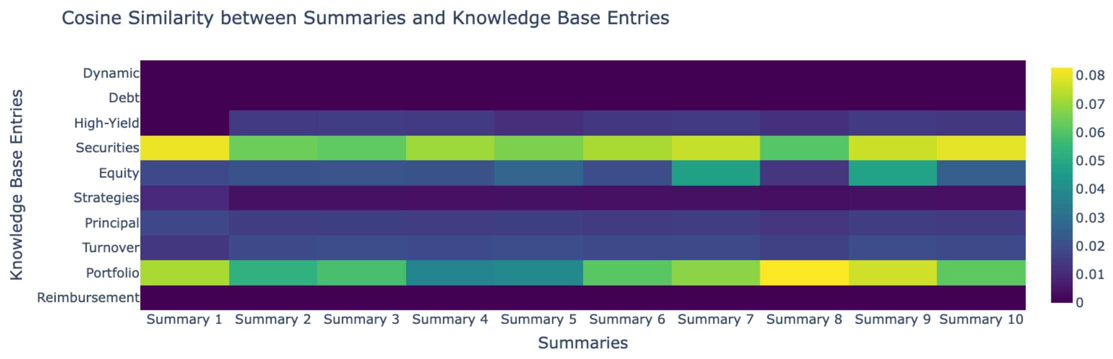


Figure 1: Knowledge Base Graph

3.2 Distance Measurement

In Figure 2, terms like "portfolio," "securities," "performance," "market," and "emerging" exhibit a higher cosine similarity between summaries and knowledge base entries when compared using the same summary lists.



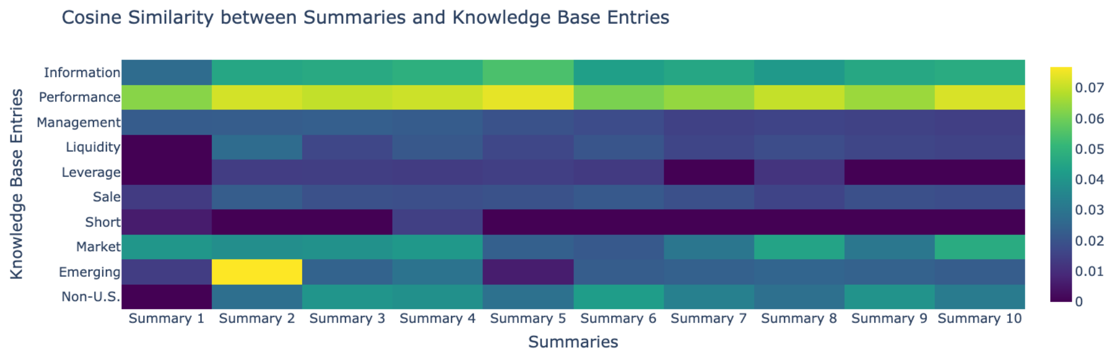


Figure 2: Cosine Similarity between Summaries and Knowledge Base Entries

3.3 Result of Building Own Word Embedding

Within the **Random Search cross-validation** to tune the validation data, the comparison between pre and post-optimization results reveals a noticeable improvement in model performance across the board in Table 1. Before optimization, XGBoost exhibited the best results, with a precision of 0.65, a recall of 0.68, and an F1-score of 0.66, while Random Forest and LightGBM trailed slightly. Post-optimization, all models showed substantial improvements, particularly Random Forest, which achieved a precision of 0.76 and an F1-score of 0.75. LightGBM and XGBoost showed significant gains as well, both reaching a precision of 0.71 and an F1-score of 0.76. These improvements underscore the importance of optimization in enhancing model accuracy, recall, and F1-score, with the optimized models consistently outperforming their pre-optimization counterparts.

3.4 Result of Using Pre-trained Model

The table 2 presents a comparison of model performance between Sentence BERT and RoBERTa using Random Forest, LightGBM, and XGBoost algorithms. The results highlight that RoBERTa consistently outperforms Sentence BERT across all three metrics: precision, recall, and F1-score. This aligns with the known characteristics of both models; RoBERTa, being a more robustly optimized variant of BERT, often demonstrates

Table 1: Model Performance Before and After Optimization

Algorithm	**Before Optimization**			**After Optimization**		
	Precision	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
Random Forest	0.61	0.65	0.63	**0.76**	0.74	0.75
LightGBM	0.64	0.66	0.65	0.71	0.81	0.76
XGBoost	**0.65**	**0.68**	**0.66**	0.71	**0.81**	**0.76**

improved performance due to enhanced pre-training strategies and a larger training dataset. Specifically, in Random Forest, RoBERTa shows the highest precision (0.80), recall (0.81), and F1-score (0.80), reflecting its effectiveness in capturing the nuances of text data. In comparison, Table 2 reveals that pre-trained models, particularly RoBERTa, provide superior performance in precision, recall, and F1-score, underscoring the value of leveraging advanced pre-trained models for better predictive outcomes compared to self-trained word embeddings.

Table 2: Model Performance of Sentence BERT and RoBERTa

Algorithm	**Sentence BERT**			**RoBERTa**		
	Precision	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
Random Forest	**0.76**	0.77	0.76	**0.80**	**0.81**	**0.80**
LightGBM	0.71	**0.81**	**0.76**	0.77	0.79	0.78
XGBoost	0.67	0.72	0.69	0.77	0.77	0.76

4 Conclusion

In conclusion, this project effectively demonstrates the power of advanced machine learning and natural language processing techniques in analyzing financial data. By using models such as Word2Vec, Sentence BERT, and RoBERTa, the study illustrates the significant impact of embedding methods on predictive model performance, enhancing the precision, recall, and F1-scores of algorithms like Random Forest, LightGBM, and XGBoost. The comprehensive analysis, which includes feature engineering, TF-IDF vectorization, and cosine similarity, highlights the importance of leveraging semantic relationships in financial texts. Additionally, models trained with self-generated embeddings show strong performance across the three classification algorithms, while the pre-trained models achieve slightly better scores overall. This demonstrates that self-trained embeddings effectively capture semantic relationships, proving their value in financial data analysis.

Contribution

I contribute 100% in this project.