AIR-mazing Predictions: Ranking of Retrieval Augmented Stock Market Prediction for Business Ideas

Kai Kainbacher - Training Model
Maximilian Legat - Evaluation, Illustration and Retrievel System
Jonathan Maier - Creating Dataset and Model Architecture
Michael Sickl - Evaluation, Documentation and Ranking System
Group 09

November 2024

Abstract

In this document we present a retrieval augmented prediction model and a ranking system that aims to predict the stock performance of a company, based on to textual description of its business idea, and rank it alongside existing competitors. The prediction and ranking is relying on how companies with similar business ideas performed on the stock market in the past. In order to achieve that, our model is being trained with business ideas and the corresponding stock performance enriched with some relevant features such as market size, investment, or team strength. Adding data of similar retrieved companies as input improves comparability and provides deeper insights into the factors contributing to success. Overall, by combining information retrieval, text-based insights and static attributes, our model aims to provide a realistic prediction of potential market success and a ranking for new business ventures.

1 Introduction

A very important aspect of starting a business, if not the most important one, is the business idea. It contains the purpose or the goal of the company. And although it is not the only relevant aspect of success, it provides the foundation for all other business procedures. Because of that, it is crucial for an upcoming entrepreneur to know whether the business idea is likely to be successful. Furthermore, if the company is present on the stock market, potential investors also have an interest in acquiring knowledge about the stock performance of a company since a more promising idea will most likely result in a greater return on their investment. Therefore, our team has decided to construct a neural network model to evaluate a business idea. Based on how existing companies with similar ideas have performed on the stock market, our model predicts how new ideas will perform and then rank them with competitors. Therefore, we address the research question:

How can textual descriptions of a new business idea be used to predict its stock performance and rank it alongside existing market competitors?

To accomplish this, we first need to gain insight into how current companies perform and what their business ideas are. Once we have that information, it is enriched with further static parameters such as market size, investments, and the size of the team to make it easier for the model to put their success into perspective. This allows the model to give a prediction of stock performance based on how similar businesses performed in the past. For a more accurate prediction, it is possible to also provide those static parameters. This document outlines how the dataset and model are constructed to suggest an accurate outlook. The source code and dataset for this project are available in our repository: `https://github.com/jonnyCap/AIR-Project`.

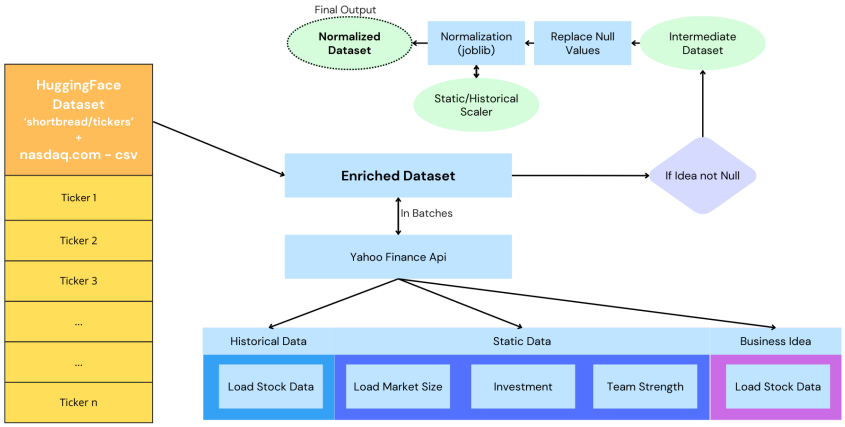
2 Dataset

The dataset used for this project was created by combining a dataset from Hugging Face with the most up to date tickers download from 'nasdaq.com'. To enrich these tickers with further information we used the Yahoo Finance API. This combination results in a hybrid dataset that integrates both static and

dynamic components and therefore can be effectively utilized to train our model. Our current dataset is to found in the Dataset/Data folder as 'normalized_real_company_stock_dataset_large.csv' in our GitHub Repository. However, due to the introduction of an Ranking System we will have to create a second Dataset, containing the business idea, the last few months of stock performance as well as a success indicator that we calculate from static information and historical performance.

2.1 Dataset Creation

The main dataset consists of both dynamic and static attributes, along with detailed textual descriptions. The dynamic data includes the historical stock performance in monthly intervals, focusing on the closing prices for the last 24 months per ticker, although this interval can be adjusted dynamically. The static attributes encompass a wide range of company-specific information, including **market size**, **investment level**, **team strength**, and additional factors such as the **sector**, **region**, **founding year**, and many other key indicators. Additionally, the dataset includes a **business description** for each company, which captures the core idea and area of activity. This is a critical component of the dataset, sourced primarily from the yahoo finance api. If the discription for a company cannot be fetched, the entry is excluded from the dataset to o maintain data integrity and avoid distortions. By combining dynamic, static, and textual data, the dataset provides a robust foundation for analysis while maintaining high data quality standards.



2.2 Data Processing

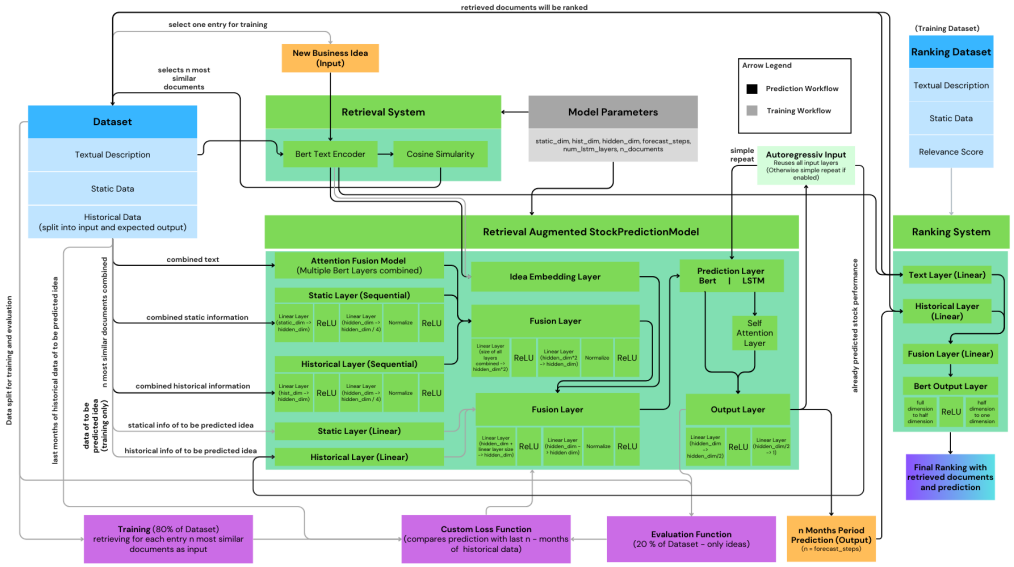
The data processing was carefully designed and further updated to handle large volumes of data efficiently. It consists of many stages to ensure correctness and prevent issues between the dataset and the model. To handle large datasets, we use **batch processing**. We start with around 7,000 stock tickers from our combined datasets and split them into smaller groups of 100 tickers each. This method helps us process the data efficiently while staying within API limits. The **validation and filtering** is performed during data collecting. This ensures that stocks with insufficient data or missing business description will be excluded. Missing values in the stock history are padded with zero, to keep the same data structure. Values that are strings were encoded using a LabelEncoder. Calculated static values that were set to infinity where replaced by the columns maximum value. After identifying the static and historical data, it has to be **normalized** in order to allow for a smooth training process. Since our data is of very different nature, we use for the static features the **StandardScaler** for each column individually, which standardizes the variables by removing the mean and scaling them to unit variance. For the dynamic variables, we use the same **MinMaxScaler** for all columns, which rescales the data to a fixed range (0,1) so that the variance is not too large.

For static features:	Normalization for dynamic features:
$z = \frac{x-\mu}{\sigma}$	$z = \frac{x-\min(x)}{\max(x)-\min(x)}$

After the dataset has been normalized, we save both the processed data and the scaler objects for static and dynamic features. By storing the scaler objects, we can ensure consistent scaling across future dataset extensions and also enable denormalization for our predictions.

3 Base Retrieval Augmented Prediction Architecture

Our system takes an idea as input, predicts its stock performance and ranks its successrate in comparison to similar retrieved companies. This is the overall workflow:



3.1 Workflow of the Retrieval-Augmented Prediction Model

The initial interaction occurs with our **Retrieval System**, which identifies companies with business descriptions similar to our new idea. This is achieved by encoding the idea and calculating the **cosine similarity** between the encoded idea and the companies in our database. The retrieved documents, combined with the encoded idea, serve as inputs for our **Prediction Model**. Incorporating the retrieved documents as inputs allows the model to gain deeper insights into the potential performance of the idea. We utilize an **Attention Model** to condense the large combined text into a meaningful embedding. This is accomplished through multiple **BERT** and **Fusion** layers. Additionally, the combined static and historical data are processed through an individual sequential layer that reduces the input dimensions. During training, additional linear input layers are used for the static and historical data of the target idea, enhancing the model's understanding. To integrate these inputs, our Prediction Model employs **two Fusion Layers**. The first layer combines the text representation generated by the Attention Model with the static and historical data. The second Fusion Layer merges the **Idea Embedding layer**, which represents the embedding of our new idea, with the output of the first Fusion Layer and the static and historical data specific to the idea (during training only). The output of the second Fusion Layer serves as input to our **Prediction Layer**, where we utilize either BERT or an LSTM in combination with an additional Attention Layer. The Prediction Layer's output is then processed by the **Output Layer**, producing a single final result. Due to the model's **autoregressive** nature, previously predicted stock performance from the output layer is reintroduced as input to the linear historical layer for subsequent predictions. This workflow is repeated iteratively until the model completes the desired number of predictions. The final array, along with the idea, is then passed to the ranking system.

3.2 Ranking system

The **Ranking System** is our subsystem that ranks both the newly predicted idea and the companies with similar business ideas retrieved by our Retrieval System at the beginning of the workflow. It consists of several layers, including the **Text Encoding Layer** and the **Historical Layer**, which serve as input layers for the textual embedding and historical data of an entry. Additionally, there is the **Fusion Layer**, which combines historical and textual data, followed by the **Output Layer**, which predicts the final score representing the success rate of an idea. Once this system is applied to the new idea, combined with its predicted stock performance, as well as the similar companies from our dataset, the entries are ranked based on the results and presented to the user. This provides the user with insights into the success rate of their idea, its potential stock performance, and how it compares to similar businesses.

4 Methods

For our three systems, we have implemented several key features to optimize our results.

4.1 Retrieval & Ranking System

In our retrieval system, the input idea is transformed into a text embedding using a **transformer-based model** like BERT. Similarly, database entries are embedded, and we calculate the **cosine similarity** between the input idea and company descriptions. For efficiency, these embedded descriptions are stored in a new dataset. The system retrieves the ten most similar documents, which serve as input for both training and predictions in our Stock Prediction Model. To enhance transparency, we plan to implement **diverse visualization techniques** to showcase the accuracy and effectiveness of our approach. Users receive documents ranked by our Ranking System, which assigns a **performance score to each document**. The performance score is generated from the company description and stock performance. Additionally, the stock prediction for their idea is ranked, enabling users to make clear comparisons. While the Retrieval System fully relies on BERT's embedding capabilities and does not require additional training, our Ranking System is trained on a dataset containing business descriptions, recent stock performance, and a score derived from various success indicators. This training ensures that the Ranking System can effectively evaluate a business based solely on its recent performance and textual descriptions.

4.2 Prediction Model

We have implemented a **custom training loop** that processes batches of the dataset for a specified number of epochs. The training data comprises 80% of the dataset, with the last 12 months of the historical data of each entry being designated as the target, while predictions are based on the first large portion of the data. This target data is essential for evaluating the model's prediction accuracy. We begin by selecting an idea from the dataset and using the Retrieval System to find similar ideas. These ideas along with statistical and historical data are used in our prediction model. The model trains on this data, prioritizing insights from the ten similar companies. If overfitting occurs, we will adopt a different approach, such as using random data for an idea or increasing the number of ideas retrieved from the retrieval system. For training, we employ a **CustomLoss** class to calculate the loss between predictions and target data. It is based on the **TemporalLoss** function, which computes temporal differences in the data and applies them as a penalty. Additionally, we incorporate mean squared error, a penalty for negative predictions, and a penalty for insufficient diversity. After each epoch, we evaluate the model's performance. The **evaluation function** takes a batch of ideas as input, representing the remaining 20% of our dataset, to assess its predictions against the target data using our custom loss function. To provide **visual feedback**, we plot the predictions alongside the target data. This graphical representation enhances our understanding of the model's performance and helps identify areas for improvement. Overall, this comprehensive approach allows us to effectively train, evaluate, and visualize our model's performance, ensuring we can refine our predictions and optimize our results.

5 Evaluation

Our evaluation process examines all systems comprehensively. We assess the retrieval system's accuracy by using visualizations to analyze cosine similarity scores. This approach enables us to visually inspect the relevance of the retrieved companies and fine-tune our embedding process if necessary. As previously mentioned, we will incorporate an evaluation method during training that visualizes the model's performance. Additionally, we will perform an in-depth analysis of the models by selecting a few new companies from the stock market, defining their core business ideas, and using our retrieval system to find similar companies. The retrieved companies and the predicted data will then be ranked to provide a detailed comparison.

6 References

Repository: `https://github.com/jonnyCap/AIR-Project`