MF815 Final Project Report

Xingqi Zhong, Ziyi Zhang, Xinyi Ying

Executive Summary

For preprocessing the data, We merge fund summary data with a column of ‘Investment Strategy’ into a dataframe. In terms of splitting the data into training and testing, we choose a random 70% of the whole dataset as training data and the rest as testing data.

After we split the data into training and testing sets, we have a summary article for each fund as our input value x, and investment strategy as our label y for later classification model. In this step, our goal is to transform our input article into vectors for later implementation by using skip-gram model.

To build a knowledge base to the according investment strategy, we use the TF-IDF scoring model to get the highest frequency words for each strategy category. Then using get n closer and create a knowledge base function to enlarge our key words to knowledge base.

After creating knowledge bases, we need to find a way to detect the distance of every summary. We first do outlier detection, then calculate scores. One way to do this is first to take the distances between the words in the knowledge base and the barycenter of the sentences. Second, quantify the distance score by averaging the N smallest distances.
After all, we can output sentences (aka top_sentences) we are going to extract by sorting first ‘num_sent’ smallest score.

Stil, there are some words not in ‘word2vec’ vectors. Therefore, we choose to delete the sentences if they are less than the number of rejected words in our ‘word2vec’ function. However, there is still some further work to be done. Because we have already created knowledge bases of each strategy type, we need to calculate the top_sentences of each fund based on different sorts of knowledge bases.

Our Result and Discussion

First issue we encountered is, we found that the number of rows of file ‘MutualFundLabels’ does not equal to that of files in ‘MutualFundSummary’. It means, there are some funds that do not have names of investment strategies. Our first idea of dealing with this issue was to try to delete data that does not have the label. However, we discussed this idea and then got the conclusion that it does not make sense assuming we delete the data. So we tried to take account of all of the data.

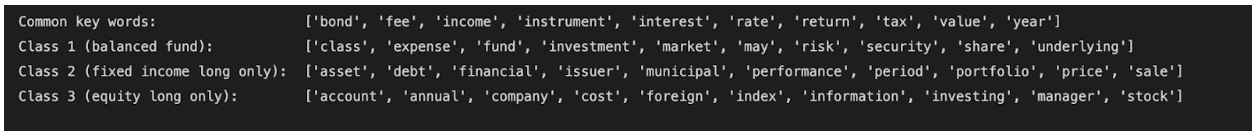
Second is, we intended to consider data without investment strategy as test data, those with investment strategy as training data. However, we realized that we need to consider None type of the strategy.

Third is, we found there exists a 5th strategy type when we applied our codes. This time we decided to delete the information that has the 5th strategy type.

After those, we all agreed that we train the amount for the training dataset as much as possible, in order to more accurately assess our final result.

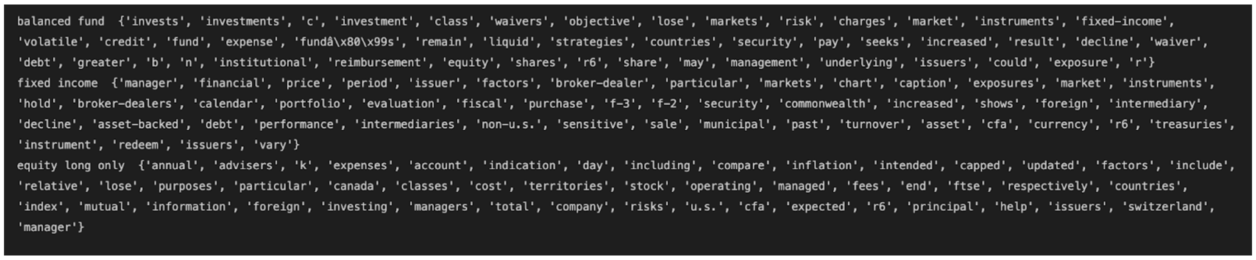
Then, move forward to constructing the knowledge base and key_words. We initially use the same stop words as input of TfidfVectorizer for each category, and only use the given stop words list. And we end up with three lists of repeating words. To improve, we try adding common key_words to stop words, and the resulting lists are still almost the same. Then we come up with the idea to put the key_words of the first category into the next to avoid the problem. However, in this method, the key words lists will be different if we change the computation sequence of categories.

key_words



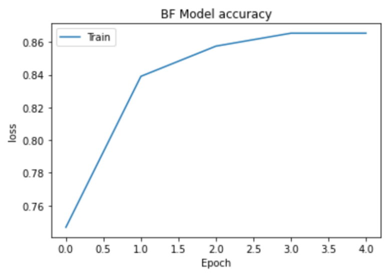
```
Common key words:                ['bond', 'fee', 'income', 'instrument', 'interest', 'rate', 'return', 'tax', 'value', 'year']
Class 1 (balanced fund):         ['class', 'expense', 'fund', 'investment', 'market', 'may', 'risk', 'security', 'share', 'underlying']
Class 2 (fixed income long only): ['asset', 'debt', 'financial', 'issuer', 'municipal', 'performance', 'period', 'portfolio', 'price', 'sale']
Class 3 (equity long only):      ['account', 'annual', 'company', 'cost', 'foreign', 'index', 'information', 'investing', 'manager', 'stock']
```

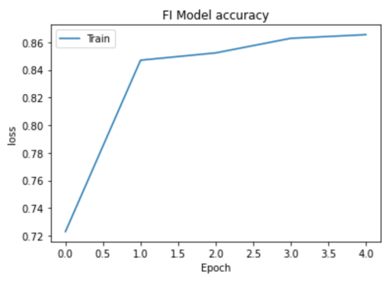
knowledge base

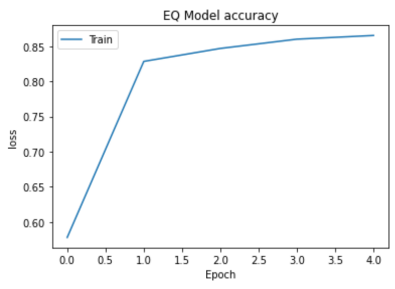


```
balanced fund  {'invests', 'investments', 'c', 'investment', 'class', 'waivers', 'objective', 'lose', 'markets', 'risk', 'charges', 'market', 'instruments', 'fixed-income',
'volatile', 'credit', 'fund', 'expense', 'fundâ\x80\x99s', 'remain', 'liquid', 'strategies', 'countries', 'security', 'pay', 'seeks', 'increased', 'result', 'decline', 'waiver',
'debt', 'greater', 'b', 'n', 'institutional', 'reimbursement', 'equity', 'shares', 'r6', 'share', 'may', 'management', 'underlying', 'issuers', 'could', 'exposure', 'r'}
fixed income  {'manager', 'financial', 'price', 'period', 'issuer', 'factors', 'broker-dealer', 'particular', 'markets', 'chart', 'caption', 'exposures', 'market', 'instruments',
'hold', 'broker-dealers', 'calendar', 'portfolio', 'evaluation', 'fiscal', 'purchase', 'f-3', 'f-2', 'security', 'commonwealth', 'increased', 'shows', 'foreign', 'intermediary',
'decline', 'asset-backed', 'debt', 'performance', 'intermediaries', 'non-u.s.', 'sensitive', 'sale', 'municipal', 'past', 'turnover', 'asset', 'cfa', 'currency', 'r6', 'treasuries',
'instrument', 'redeem', 'issuers', 'vary'}
equity long only  {'annual', 'advisers', 'k', 'expenses', 'account', 'indication', 'day', 'including', 'compare', 'inflation', 'intended', 'capped', 'updated', 'factors', 'include',
'relative', 'lose', 'purposes', 'particular', 'canada', 'classes', 'cost', 'territories', 'stock', 'operating', 'managed', 'fees', 'end', 'ftse', 'respectively', 'countries',
'index', 'mutual', 'information', 'foreign', 'investing', 'managers', 'total', 'company', 'risks', 'u.s.', 'cfa', 'expected', 'r6', 'principal', 'help', 'issuers', 'switzerland',
'manager'}
```

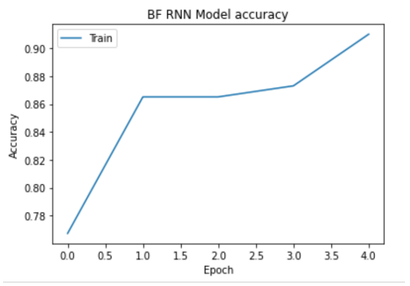
Aftering deciding to try two algorithms to finish our project, we construct 3 CNN models based on the three different knowledge bases as below. We can easily see that CNN model has the best fitting result on the ‘Balance Fund’ knowledge base and ‘FIxed Income Long Only’ knowledge base, and finally on the ‘Equity Long Only’ knowledge base.
One of the problems we encounter in training CNN models is overfitting. As we train the CNN model with given layers and parameters, the result shows that the accuracy doesn’t improve much after training the first batch. This indicates we have an overfitting problem, we tried several methods to improve the results.
Our first trial is to tune the model hyperparameters. The model parameters epoch number and batch size were initially set to 10 and 16, and by changing them into 5 and 100. Enlarging the batch size allows us to lower the learning rate.
In the second method, we want to simplify the model by removing some layers or adding the L2 regularizer. L2 Regularizer will penalize complex models and thus force it to be simpler. They both successfully slow our model’s learning rate as we can see in the graph of CNN model accuracy.

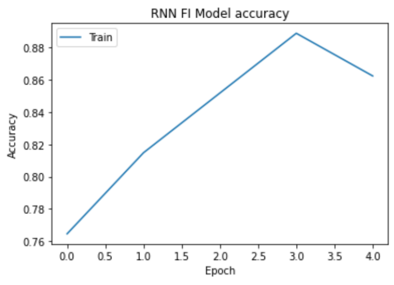


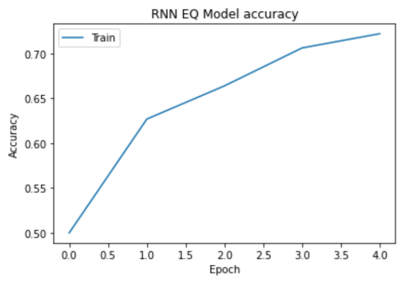




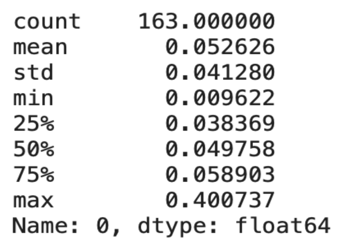
We also construct 3 RNN models based on the three different knowledge bases as below. We can easily see that the RNN model has the best fitting result on the ‘Balance Fund’ knowledge base, then on the ‘FIxed Income Long Only’ knowledge base, and finally on the ‘Equity Long Only’ knowledge base.





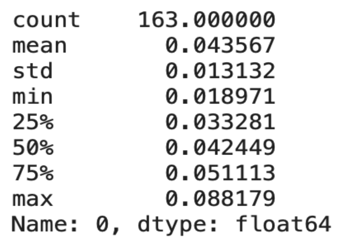


For classification of the funds in the corresponding three categories, the description for the prediction result for the probability of belonging to the first category is shown below.



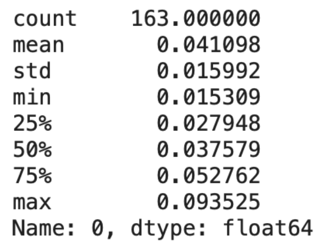
```
count    163.000000
mean       0.052626
std        0.041280
min        0.009622
25%        0.038369
50%        0.049758
75%        0.058903
max        0.400737
Name: 0, dtype: float64
```

CNN Prediction for Balanced Fund



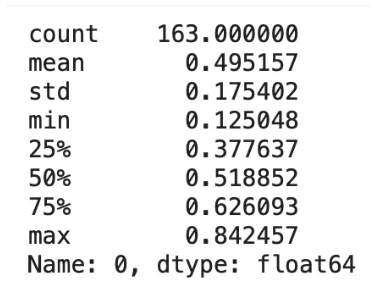
```
count    163.000000
mean       0.043567
std        0.013132
min        0.018971
25%        0.033281
50%        0.042449
75%        0.051113
max        0.088179
Name: 0, dtype: float64
```

CNN Prediction for Fixed Income Long Only



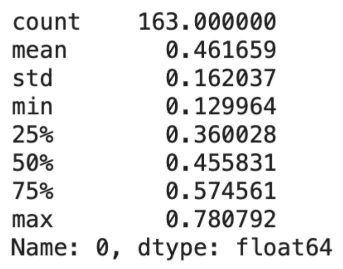
```
count    163.000000
mean       0.041098
std        0.015992
min        0.015309
25%        0.027948
50%        0.037579
75%        0.052762
max        0.093525
Name: 0, dtype: float64
```

CNN Prediction for Equity Long Only



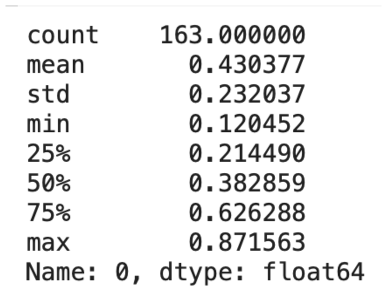
```
count    163.000000
mean       0.495157
std        0.175402
min        0.125048
25%        0.377637
50%        0.518852
75%        0.626093
max        0.842457
Name: 0, dtype: float64
```

RNN Prediction for Balanced Fund



```
count    163.000000
mean       0.461659
std        0.162037
min        0.129964
25%        0.360028
50%        0.455831
75%        0.574561
max        0.780792
Name: 0, dtype: float64
```

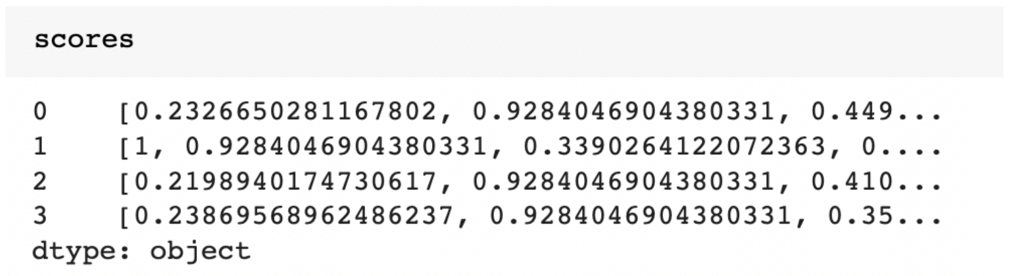
RNN Prediction for Fixed Income Long Only



```
count    163.000000
mean       0.430377
std        0.232037
min        0.120452
25%        0.214490
50%        0.382859
75%        0.626288
max        0.871563
Name: 0, dtype: float64
```

RNN Prediction for Equity Long Only

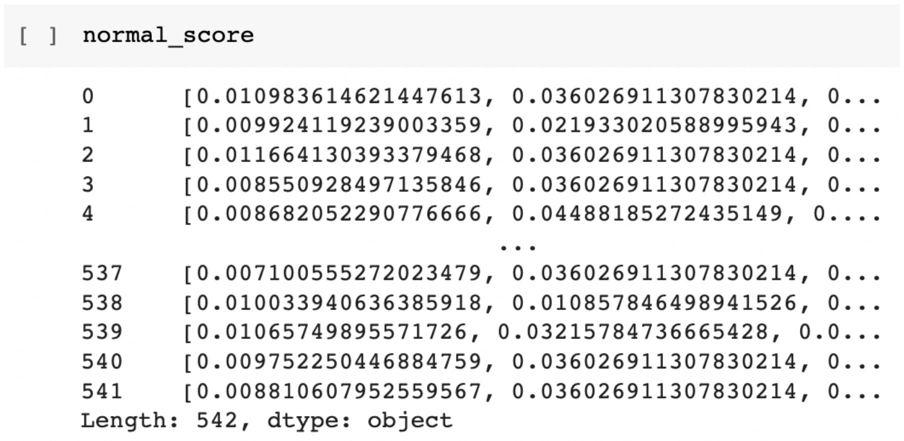
For classification of the fourth category, we think it unrealistic to construct a knowledge base using four sample points. We applied the outlier detection method to find outliers in the distance of the encoded vectors of fund summaries to the three knowledge bases. We calculated the scores of the funds from the fourth category with respect to the knowledge base of the first category, which represents the average distance of the five sentences that are closest to the knowledge base, as is shown below.



```
scores

0    [0.2326650281167802, 0.9284046904380331, 0.449...
1    [1, 0.9284046904380331, 0.3390264122072363, 0....
2    [0.2198940174730617, 0.9284046904380331, 0.410...
3    [0.23869568962486237, 0.9284046904380331, 0.35...
dtype: object
```

We also computed the scores of the other funds to the same knowledge base.



```
[ ]  normal_score

     0      [0.010983614621447613, 0.036026911307830214, 0...
     1      [0.009924119239003359, 0.021933020588995943, 0...
     2      [0.011664130393379468, 0.036026911307830214, 0...
     3      [0.008550928497135846, 0.036026911307830214, 0...
     4      [0.008682052290776666, 0.04488185272435149, 0....
                                         ...
     537    [0.007100555272023479, 0.036026911307830214, 0...
     538    [0.010033940636385918, 0.010857846498941526, 0...
     539    [0.01065749895571726, 0.03215784736665428, 0.0...
     540    [0.009752250446884759, 0.036026911307830214, 0...
     541    [0.008810607952559567, 0.036026911307830214, 0...
     Length: 542, dtype: object
```

As we can see from the figures, the scores of the funds from the fourth category is significantly larger than that of the other funds. Thus, we can conclude that those funds are from the fourth category.

Methodology

Randomly selecting
As Executive Summary mentioned before, we randomly choose 70% of the whole dataset as a training set and the rest as a test set.

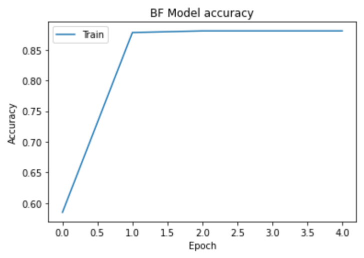
Skip-gram model
The skip gram model is a three layers deep learning model. This model takes in a set of words and assigns each word a unique vector, which allows words in the same context to be close to each other.
It works with the following steps. First it takes in a word and predicts its neighboring word. Next remove the output layer and with only the input and hidden layer, we input a vocabulary and in the layer is our ‘word embedding’ of input.

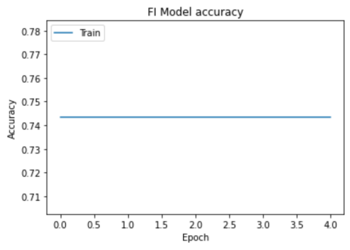
TF-IDF
To transform words into vectors, we use TF-IDF, which is a scoring algorithm that takes in a word and outputs the score of frequency. The scoring algorithm contains two parts, first is the IDF scoring. The scoring function for IDF is the total number of the articles in the corpus over the number of articles with the word in it and take the logarithm. IDF scores represent the level of the rareness in the corpus.
The second part is the TF scoring. TF score represents how frequent this word appears in one article, and by multiplying TF score with the IDF score, this TF-IDF assigns higher score for a word that is considered special with less frequency in the whole corpus.

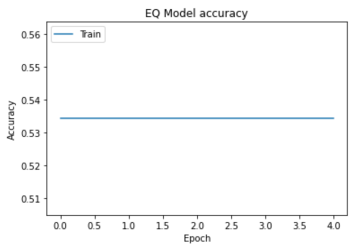
In terms of classification, we choose 2 algorithms: CNN, RNN.

CNN
In terms of choosing CNN, we train the model to optimize the filters that are implemented on the vectors of the words. In this process, we worry about overfitting the CNN model. We can adjust parameters to fit the model: detecting the difference between the accuracies of the train set and those of the test set.

Before we construct, we did fit some relatively bad models as follows. One way to solve this problem is to adjust parameters in the models.







RNN
In terms of choosing RNN, RNN layers extracts every word orderly, and on a given word, builds a vector that combines the information of the current word with the information of all previous words. As the layer goes through all the documents orderly. So RNN tends to find the best way to understand the sequence of words. In order to avoid overfitting as much as possible, we also detect the difference between the accuracies of the train set and those of the test set.

Outlier detection
Since the fourth category only has four samples, it's unrealistic to predict this category by splitting it into a train and test set and fitting a CNN or RNN model using the training set. Thus, we calculated the distance of the funds to the three knowledge bases which have been constructed in the previous steps. And then we used the LocalOutlierFactor function in sklearn to detect the outliers in the list of distances. Those with outlying distances should belong to the fourth category, i.e. Long Short Funds (High Risk).

Appendix

To transform our input article into vectors:
First step is to process our text as input. We start by splitting text into separate words, then filtering out the meaningless words. We use the function tokenizer in our code to do the split, normalize and filtering. Then, to apply the skip-gram model, we specify some training parameters, such as the max vocabulary number and the min occurrence number. Next step is to further process our input by building a dictionary. We filter words by setting upper bound and lower bound of occurrence frequency and assigning a unique id to each of the filtered words. With the filtered words as key and unique id as value, we build a dictionary. Then, using the

dictionary, we can get the unique id from the index, and assign words with low occurrence and not in the dictionary with value 0. And we build a vector with value 0 and replace it with 1 where there is an id at that position.
Then we use the batch generator function to generate a new batch of inputs with our data and train the built-up skip-gram model with a batch generator.

To build a knowledge base:
Since there are four categories with one considered as outlier, we decide to build three knowledge bases without the high risk fund.
First, we set the number for the max feature, which is the number of top used words. In our project, we set it to 10. Create the TFIDF object using the previous function of lemma tokenizer function and stop words list. Then, fit the TFIDF vectorizer on the training sample and create the training features. Then we can get the 10 most frequently used words in summary.
To get key words for each three categories, we need to filter out a summary of each investment strategy category. We build a data frame with funds name and summary and merge the label file on funds name. And by further merging on each investment strategy, we now have three data frames.
To avoid common keywords appearing in the keywords list for each category, we decided to include the common keywords into the stop words in the TfidfVectorizer function. And adding the resulting keywords for the first category to stop words used in the second one. And by repeating this process, we have three sets of key words without common words.
Then apply the get_n_close and create_knowledge_base function to get knowledge base.

Who have done what in this project

Task1 split data set : Xingqi Zhong
Task2 word embedding dictionary : Ziyi Zhang
Task3 knowledge base: Ziyi Zhang, Xingqi Zhong
Task4 measure distance/ classification/outlier detection: Xingqi Zhong, Xingyi Yin(classification on the 2 knowledge base)
Task5 predict investment strategy Xingyi Yin, Xingqi Zhong

Xingqi Zhong: Task 1, 4, 5 code part. Work with Ziyi on Task 3. Write outlier part on Paper.

Xingyi Yin: Code part on Task 5 and Part of Task 4. Write report on Task 1 4 5

Ziyi Zhang: Task 2 &3 code part. Write word embedding and knowledge base part on Paper.