MF 815
Final Project
Weiliang Gu & Haoyang Shang
Prof. Hao Xing

Mutual Fund Strategy NLP Classification Project

Part I: Data Processing

In this project, we were given a collection of mutual fund summaries and a CSV file containing the investment strategy of each fund. Our goal was to develop an NLP model to predict which strategy each fund uses based on the given text summaries. We were given a total of 545 fund summaries and the csv file containing the strategies of 467 funds. Therefore, we matched the fund summaries with the labels and set the missing entry of funds as our potential testing set. The test dataset contained 79 funds that did not have a label but was in the folder of summaries.

Immediately afterward, we would like to see how many strategies there were in the CSV label column. Using the count command, we got the following information.

```
df_label['Ivestment Strategy'].value_counts()
```

```
Equity Long Only (Low Risk)          248
Fixed Income Long Only (Low Risk)    130
Balanced Fund (Low Risk)              84
Long Short Funds (High Risk)           4
Commodities Fund (Low Risk)            1
Name: Ivestment Strategy, dtype: int64
```

We saw that there were five strategies in total, despite one strategy only having 4 cases and one other strategy with a single entry. The sample size for those two strategies was too small for training a valid model, and realistically speaking we couldn't even guarantee that the random

train-test splitter would put the cases into the training set. However, we still tried to train two models: one with 3 classes of Investment Strategy and one with 4 classes of Investment Strategy.

We will call them Model3 and Model4 in the following passage.

We set the train-test split ratio at 0.3 and checked that there were “Long Short Funds” in both train and test dataset.

Part II: Skip-Gram Model

We feed the training set data into the Skip-Gram Model and obtained the eighteen most common keywords from summaries that describe the investment strategy for both Model3 and Model4. The keywords were as follows:

```
['class', 'company', 'expense', 'fee', 'fund', 'investment', 'market', 'may', 'performance', 'portfolio', 'rate', 'return', 'ri
sk', 'security', 'share', 'tax', 'value', 'year']
['class', 'company', 'expense', 'fee', 'fund', 'index', 'investment', 'market', 'may', 'portfolio', 'rate', 'return', 'risk',
'security', 'share', 'tax', 'value', 'year']
```

Part III: Knowledge Base

Upon investigation, we believed that these keywords make perfect sense, and we went further ahead to create the knowledge base using the five closest words to each keyword. After removing repetitive words, we get the following knowledge base. Again, for both Model3 and Model4:

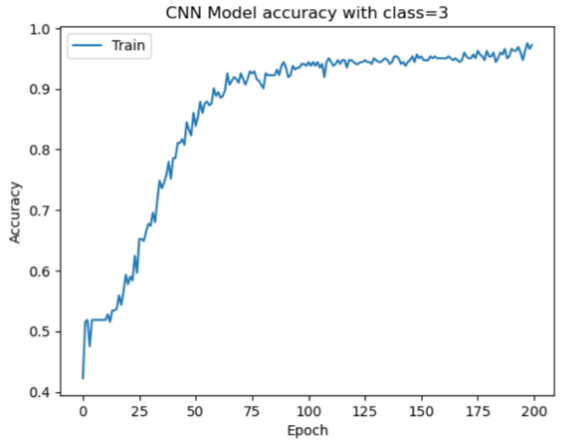
```
{'asset-based', 'inopportune', 'indices', 'receives', 'determines', 'small-capitalization', 'owned', 'depicts', 'engage', 'extr
emely', 'customers', 'heightened', 'may', 'assurance', 'beginning', 'accounts', 'committee', 'unanticipated', 'share', 'withdra
wn', 'commencement', 'track', 'platforms', 'waiving', 'remainder', 'impaired', 'class', 'portfolio', 'fee', 'contacting', 'ordi
narily', 'expenses1', 'borne', 'sixth', 'r', 'return', 'billion', 'risk', 'described', 'minimum', 'rate', 'remains', 'difficul
t', 'entirely', 'updated', 'customarily', 'demand', 'loss', 'r-3', 'sooner', 'tax', 'effect', 'commercial', 'developments', 'co
mpetitive', 'usaa.com', 'etf', 'transact', 'attempt', 'estate-linked', 'investment', 'ownership', 'company', 'creditworthines
s', 'strategy', 'pacific', 'security', 'year', 'b', '2a-7', 'concentrates', 'market', 'reinvestment', 'value', 'deteriorating',
"'s", 'sell', 'factors', 'e', 'fund', 'relative', 'model', 'effects', 'performance', 'medium', 'documents', 'must', 'expense',
'majority'}
{'asset-based', 'inopportune', 'indices', 'receives', 'determines', 'small-capitalization', 'owned', 'depicts', 'engage', 'extr
emely', 'customers', 'heightened', 'may', 'assurance', 'beginning', 'accounts', 'committee', 'unanticipated', 'share', 'withdra
wn', 'commencement', 'track', 'platforms', 'waiving', 'remainder', 'impaired', 'class', 'portfolio', 'fee', 'contacting', 'ordi
narily', 'expenses1', 'borne', 'sixth', 'r', 'energy', 'return', 'billion', 'risk', 'described', 'minimum', 'rate', 'remains',
'difficult', 'entirely', 'floaters', 'updated', 'customarily', 'demand', 'exhibit', 'loss', 'r-3', 'sooner', 'tax', 'commercia
l', 'developments', 'competitive', 'etf', 'transact', 'attempt', 'estate-linked', 'obsolescence', 'investment', 'ownership', 'c
ompany', 'creditworthiness', 'strategy', 'pacific', 'security', 'year', 'b', '2a-7', 'concentrates', 'market', 'reinvestment',
'value', 'deteriorating', "'s", 'sell', 'index', 'factors', 'e', 'fund', 'relative', 'model', 'effects', 'medium', 'documents',
'expense', 'majority'}
```

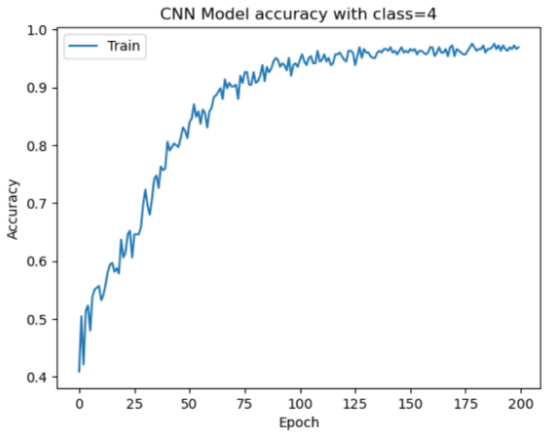
Part IV: Measure Sentence Distance

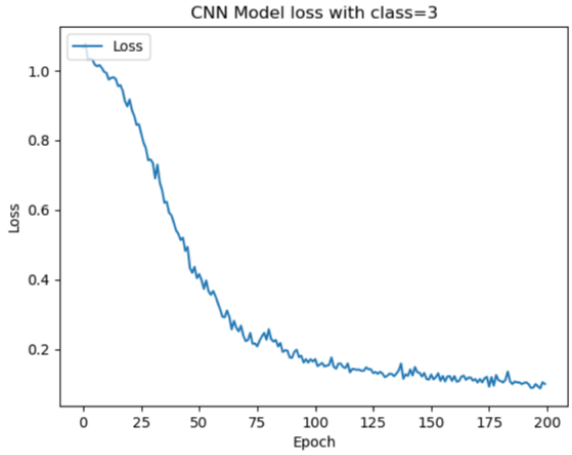
After acquiring the knowledge base, we measured the distance of sentences to the knowledge base to train our model. We chose 2500 words as the size for the vocabulary and 150 words from each document for training to achieve the best result (using empirical values from NLP lecture notes).

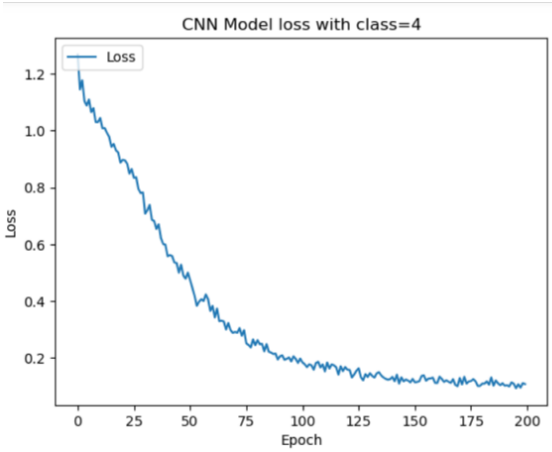
Part V: CNN & RNN Model

We trained the model using both CNN and RNN to compare their performance. First, we tested CNN using 200 epochs and a batch size of 100 on both Model3 and Model4.









After plotting the accuracy and loss of the CNN model, we could see that both values converged but did not keep constant. Since CNN was not a linear model, this sort of performance was consistent with our expectations. Next step, we want to see its performance using the validation set data.

0.6834532374100719

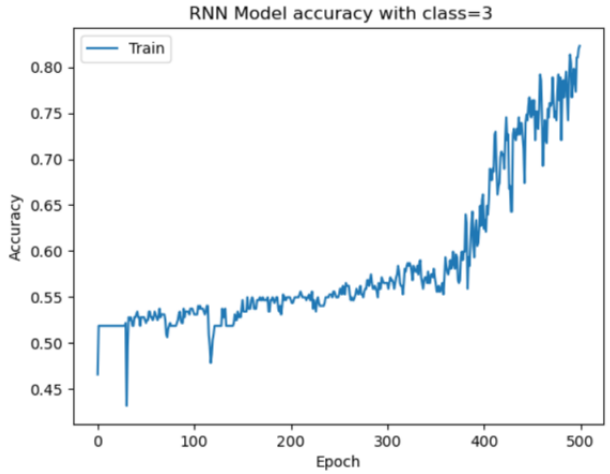
	precision	recall	f1-score	support
0	0.60	0.52	0.56	23
1	0.72	0.79	0.75	80
2	0.62	0.56	0.59	36
micro avg	0.68	0.68	0.68	139
macro avg	0.65	0.62	0.63	139
weighted avg	0.68	0.68	0.68	139
samples avg	0.68	0.68	0.68	139

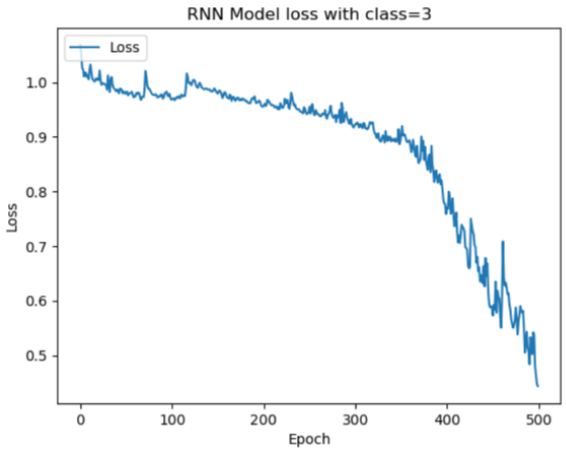
0.6928571428571428

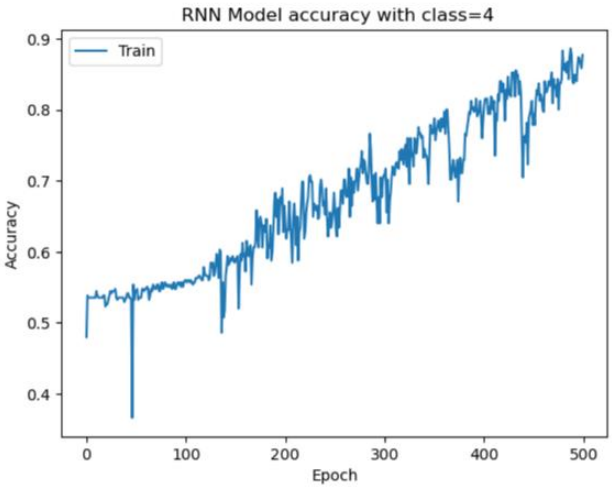
	precision	recall	f1-score	support
0	0.65	0.71	0.68	24
1	0.71	0.81	0.76	73
2	0.68	0.49	0.57	43
3	0.00	0.00	0.00	0
micro avg	0.69	0.69	0.69	140
macro avg	0.51	0.50	0.50	140
weighted avg	0.69	0.69	0.69	140
samples avg	0.69	0.69	0.69	140

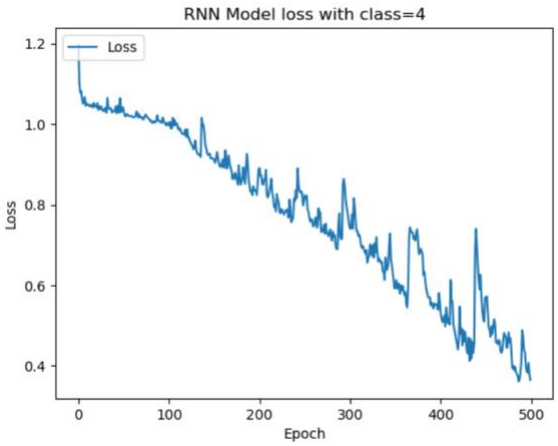
As shown in the graph, the CNN model reached an accuracy score of 0.68 for Model3 and 0.69 for Model 4. and the f1-score for the Model3 were 0.56, 0.75, 0.59 and for the Model 4 were 0.68,0.76,0.57,0.

Similarly, we trained the RNN model using a batch size of 64 and 500 epochs









We saw huge spikes in both accuracy and loss, which could potentially be explained by the depth of neural networks. Therefore, for RNN, it is crucial that we kept the epoch size large and kept checking its accuracy and loss. We investigated RNN model accuracy as the following reports:

0.6258992805755396

	precision	recall	f1-score	support
0	0.57	0.74	0.64	23
1	0.76	0.66	0.71	80
2	0.44	0.47	0.45	36
micro avg	0.63	0.63	0.63	139
macro avg	0.59	0.62	0.60	139
weighted avg	0.64	0.63	0.63	139
samples avg	0.63	0.63	0.63	139

0.6285714285714286

	precision	recall	f1-score	support
0	0.53	0.67	0.59	24
1	0.73	0.67	0.70	73
2	0.53	0.53	0.53	43
3	0.00	0.00	0.00	0
micro avg	0.63	0.63	0.63	140
macro avg	0.45	0.47	0.46	140
weighted avg	0.64	0.63	0.63	140
samples avg	0.63	0.63	0.63	140

We can see with our default parameters setting, the CNN beat RNN. But this may change if we tune parameters for both models.

Part VI: Parameters Tuning

From the plotting of accuracy and loss for both CNN and RNN, we can see their speeds to convert are not fast. So, we decided to enlarge epoch, hoping to find better hyper parameters.

We set the epoch at 800 for both CNN and RNN model and here are the reports after that:

CNN Model 3:

0.6474820143884892

	precision	recall	f1-score	support
0	0.57	0.70	0.63	23
1	0.77	0.68	0.72	80
2	0.49	0.56	0.52	36
micro avg	0.65	0.65	0.65	139
macro avg	0.61	0.64	0.62	139
weighted avg	0.66	0.65	0.65	139
samples avg	0.65	0.65	0.65	139

CNN Model 4:

0.7142857142857143

	precision	recall	f1-score	support
0	0.70	0.79	0.75	24
1	0.72	0.79	0.75	73
2	0.72	0.53	0.61	43
3	0.00	0.00	0.00	0
micro avg	0.71	0.71	0.71	140
macro avg	0.53	0.53	0.53	140
weighted avg	0.71	0.71	0.71	140
samples avg	0.71	0.71	0.71	140

RNN Model 3:

0.7194244604316546

	precision	recall	f1-score	support
0	0.65	0.65	0.65	23
1	0.80	0.80	0.80	80
2	0.58	0.58	0.58	36
micro avg	0.72	0.72	0.72	139
macro avg	0.68	0.68	0.68	139
weighted avg	0.72	0.72	0.72	139
samples avg	0.72	0.72	0.72	139

RNN Model 4:

0.6642857142857143

	precision	recall	f1-score	support
0	0.53	0.83	0.65	24
1	0.78	0.63	0.70	73
2	0.64	0.63	0.64	43
3	0.00	0.00	0.00	0
micro avg	0.66	0.66	0.66	140
macro avg	0.49	0.52	0.49	140
weighted avg	0.69	0.66	0.67	140
samples avg	0.66	0.66	0.66	140

From the results above, the best model should be the RNN Model3, with 0.72 accuracy score and f1-scores are 0.65,0.80,0.58. We suppose this should be a reasonably good result.

There were also several factors that we took into consideration while tuning the neural networks, such as train-test split ratio, number of nearest words, keywords length, and knowledge base size. We chose 150 words from each summary document as it outperforms 50, 100, 140, 160, and 200 from our testing. It turns out 150 performs best. But limited to time and computational power, we were not able to search the best for all of them. This may be the future direction for this problem.

Part VI: Conclusion and Prediction

From our discussion above, we think it is a better choice to exclude the “Long Short Funds” class to get a better result, which coincides with our initial hypothesis.

After obtaining the best model, which was RNN Model3, we implement it on the test set data that was previously mentioned. We made predictions on which of the three strategies each mutual fund using RNN Model3.

Weiliang Gu was responsible for writing parts 1-4 of the code and the first half of the report. Haoyang Shang was tasked with writing the latter half of the two.

Appendix

This project used part of Prof. Hao Xing's code from lecture on NLP Case Study.