Yuan CHEN
50015990

Summary Report of DSAA5002 Project

1. Introduction

This report will initially describe my thoughts on how to complete the project. As the Introduction part, I will make a flowchart of my complete solution idea into this part so that you can understand my general idea of solving the problem.

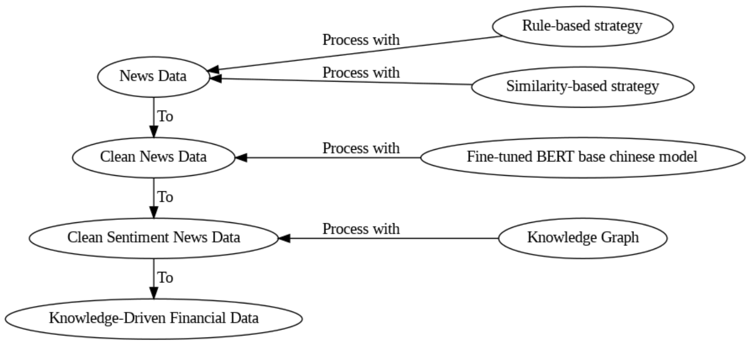

Rule-based strategy
Process with
News Data
Process with
Similarity-based strategy
To
Clean News Data
Process with
Fine-tuned BERT base chinese model
To
Clean Sentiment News Data
Process with
Knowledge Graph
To
Knowledge-Driven Financial Data

Figure 1. Complete workflow of the project

Also, for the sake of completeness, I'll fully describe my main ideas (with as little involvement as possible in the strategies I didn't use) in the second & third sections. Also, I will discuss the strategies I tried but did not use in the "Discussion" section.

2. Task 1: Data Preprocessing and Analysis

2.1 Data Preprocessing - Noise Removal

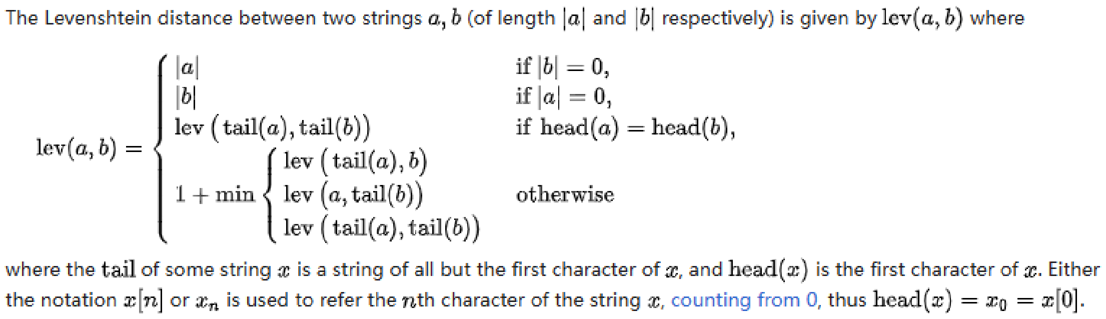
The objective of this part is to find the rows of data that contain the name of A-share company and extract the stock name. In order to achieve this goal, it is necessary to specify the existence of matching objects and the form of existence of the stock name in the news. For the matching object, my strategy is to use both the Title column and NewsContent column of the data as the matching object to maximize the completeness of the result after matching. As for the existence form, I think there are three main types: the full stock name, other formal representations of the company (fullname and code), and variants. In order to match out all three parts at the same time, I chose to consider the use of **both rule matching and similarity**.

Based on rule matching, I created a list containing both stock name, fullname, and code to match. And created a dictionary mapping the fullname and the code to its stock name to make it easier to populate the name into the subsequent Explicit_Company column. This approach primarily ensured that I had an accurate match for the first two of the name's forms of existence (the full stock name, other formal representations of the company). For this section I also did some processing of the fullname and code. For fullname, I eliminated the '股份有限公司' suffix to minimize match time loss while maintaining match accuracy. As for the code, since there is no code suffix or the suffix is in front of the code in some of the news, I removed the '.SZ' and '.SH' from the code and just kept the numbers. This still ensures match accuracy.

Table 1. Type of code error

Type of error	NewsContent (Part)
No code suffix	如（600270）外运发展，该股拥有显著的盈利能力…
The suffix is in front of the code	依米康（SZ 300249，收盘价：10.38 元）发布公告称…

For similarity, I did not choose an operational approach based on text vectorization. Instead, the NER model was first used to identify the ORG (organization name) in the input text. For similarity, I expect it to recognize variants. I did not choose an operational approach based on text vectorization. Instead, the NER model (LAC Model) was first used to identify the ORG (organization name) in the input text. Then the identified organization names are matched with all the company names (name) by doing Levenshtein Distance, and for each organization name, the one matching company name with the highest score and above the threshold of 90 is retained. Here I have added a layer of subset judgment before entering the company name in order to minimize the occurrence of matching errors. That is, the result is retained only if the matched stock name is a literal subset of the organization name or if the organization name is a literal subset of the stock name. This further reduces the probability of mismatches.


The Levenshtein distance between two strings a, b (of length $|a|$ and $|b|$ respectively) is given by $\operatorname{lev}(a,b)$ where

$$\operatorname{lev}(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ \operatorname{lev}\big(\operatorname{tail}(a), \operatorname{tail}(b)\big) & \text{if } \operatorname{head}(a) = \operatorname{head}(b), \\ 1 + \min \begin{cases} \operatorname{lev}\big(\operatorname{tail}(a), b\big) \\ \operatorname{lev}\big(a, \operatorname{tail}(b)\big) \\ \operatorname{lev}\big(\operatorname{tail}(a), \operatorname{tail}(b)\big) \end{cases} & \text{otherwise} \end{cases}$$

where the tail of some string x is a string of all but the first character of x, and $\operatorname{head}(x)$ is the first character of x. Either the notation $x[n]$ or x_n is used to refer the nth character of the string x, counting from 0, thus $\operatorname{head}(x) = x_0 = x[0]$.

Figure 2. Levenshtein distance (From Wikipedia)

In addition to this, due to the weak identification accuracy of the LAC model for some firm-like organizations. (In fact, it is only relatively weak. Its identification ability is already stronger than many NER models, and NER models with stronger identification ability than it have a significant disadvantage in computational speed compared to LAC. For this part, will discuss in Discussion section). So, I adjusted the computational threshold of Levenshtein Distance to 90. Also, to get the best possible results, I found and added some abbreviations to the rule-based matching dictionary in advance and adding some rules to skip some ORGs before calculating the similarity that are easy to wrong-match.

I wrote all of the above as functions and used .apply() to call the operations on the Dataframe directly and used the set format to ensure that there were no duplicate company names in the output data (Dataframe output operations in subsequent problems are preprocessed with set() to prevent duplicate items). The final $filter\ rate$ is as follows.

$$\text{filter rate} = \frac{527186}{1037035} \approx 50.836\%$$

2.2 Data Analysis - Text Knowledge Mining

The core of this part of the task is recognizing news sentiment. In this section, I tried two main approaches: sentiment analysis based on an existing Chinese sentiment analysis library and sentiment analysis using an autonomously fine-tuned language model. In order to compare the two methods, I collaborated with others and manually labeled a portion of news data (379 items) for the fine-tuning and testing of the model results. After the test, the latter of the two methods is the one I finally applied.

For sentiment analysis based on an existing Chinese sentiment analysis library. I used the SnowNLP model, which can be called directly in python, and did not use NLTK. the model obtained 79.82% correctness in the prediction of the test set with my manually labeled data. After I did the fine-tuning

by my own labeled training set, the correct rate became 24.56%. The results were even worse.
For the second approach, sentiment analysis was performed using the autonomously fine-tuned language model. I chose to use Bert Base Chinese with fine-tuning for the prediction of the test set, and the correct rate of the model after fine-tuning was 92.11%. And the confusion matrix of the model is obtained as follows.

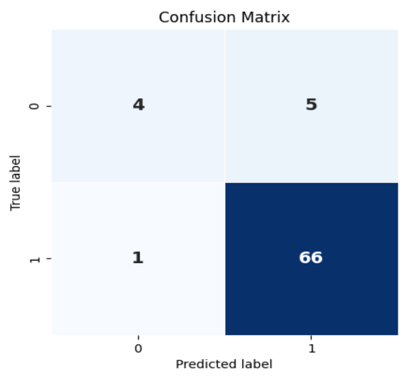


Figure 3. Sentiment Prediction Confusion Matrix of Fine Tuning Model

The above figure shows that the modeling results are very good. This is because when labeling the data, we labeled all the relatively neutral news with 1, while 0 labels are actually only about 15% in the entire labeled dataset (11% in the test set). Thus the focus of the model i.e., while identifying as many 0-labels as possible, minimizes the number of data that are truly 1-labels and are predicted to be 0-labels (i.e., the number of FNs). At above figure shows, the fine-tuned model made only 1 time such mistake, which was effective. In order for the model to learn as much as possible about 0 labels, I gave 8 times more weight to 0 labels than 1 labels when calculating the loss during model training. (For this step I performed a grid search with weights ranging from 5 to 10 at intervals of 1.)

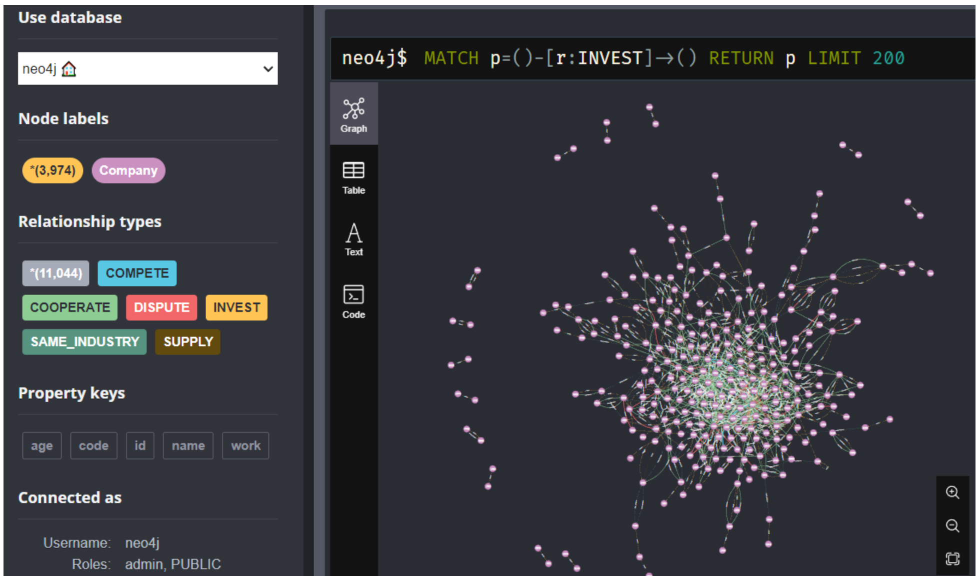
3. Task 2: Application of Knowledge Graph

3.1 Constructing a Knowledge Graph

The focus of this step is to build a reasonable knowledge graph on Neo4j. I will use the py2neo library to connect to the local neo4j Desktop and write commands using Cython statements.
During the build process I chose to use the following code to ensure that I could fully build all the relationships in the relational data, including the self-connections (10 in total) and the case where the START_ID and END_ID swapped places. It also ensures that duplicate relationships are removed.

```
LOAD CSV FROM 'file:///hidy.relationships.{rel}.csv' AS row
WITH row
SKIP 1
MATCH (start:Company {{id: row[0]}})
MATCH (end:Company {{id: row[1]}})
MERGE (start)-[r:{rel_type}]->(end)
```

Relationships created through the code as above are simply passed through the MATCH (start:Company)-[r]-(end:Company) command, which returns the bidirectional relationship for use in the next step.
The model outputs the following knowledge graph diagram.



```
neo4j$ MATCH p=()-[r:INVEST]→() RETURN p LIMIT 200
```

Use database: neo4j

Node labels: *(3,974) Company

Relationship types: *(11,044) COMPETE COOPERATE DISPUTE INVEST SAME_INDUSTRY SUPPLY

Property keys: age code id name work

Connected as

Username: neo4j
Roles: admin, PUBLIC

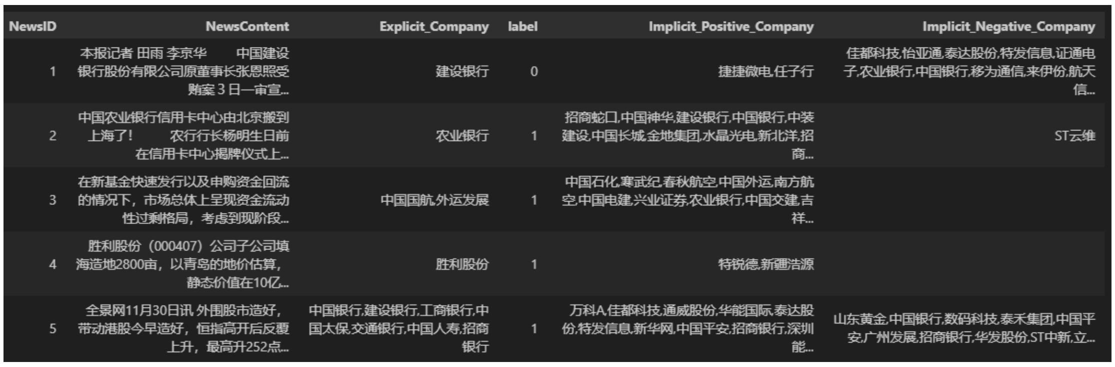
Figure 4. The knowledge graph

3.2 Knowledge-Driven Financial Analysis

For this section, I built a search-add mechanism based on the graph query approach. I chose to utilize code, which is the stock code, for matching because the stock names in the A_share_list.json file and the stock names in hidy.nodes.company.csv may differ due to a change in the stock's status, but the stock code code is a unique identifier and will not be duplicated. So I use the code in both the lists for matching.

My main idea is as follows: first make A_share_list.json connected to the knowledge graph built in Q3 by code to return the ID corresponding to the stock, then use the ID to return all its relationships as START_ID from the knowledge graph (this step uses MATCH (start:Company)-[r]-(end. Company) command to get the bi-directional relationship results), and finally populate the data with the results based on the returned information and conditions.

The first five lines of the final result obtained are output as follows.



NewsID	NewsContent	Explicit_Company	label	Implicit_Positive_Company	Implicit_Negative_Company
1	本报记者 田雨 李京华　　中国建设银行股份有限公司原董事长张恩照受贿案 3 日一审宣...	建设银行	0	捷捷微电,任子行	佳都科技,怡亚通,泰达股份,特发信息,证通电子,农业银行,中国银行,移为通信,来伊份,航天信...
2	中国农业银行信用卡中心由北京搬到上海了!　　农行行长杨明生日前在信用卡中心揭牌仪式上...	农业银行	1	招商蛇口,中国神华,建设银行,中国银行,中装建设,中国长城,金地集团,水晶光电,新北洋,招商...	ST云维
3	在新基金快速发行以及申购资金回流的情况下，市场总体上呈现资金流动性过剩格局，考虑到现阶段...	中国国航,外运发展	1	中国石化,寒武纪,春秋航空,中国外运,南方航空,中国电建,兴业证券,农业银行,中国交建,吉祥...	
4	胜利股份（000407）公司子公司填海造地2800亩，以青岛的地价估算，静态价值在10亿...	胜利股份	1	特锐德,新疆浩源	
5	全景网11月30日讯 外围股市造好，带动港股今早造好，恒指高开后反覆上升，最高升252点...	中国银行,建设银行,工商银行,中国太保,交通银行,中国人寿,招商银行	1	万科A,佳都科技,通威股份,华能国际,泰达股份,特发信息,新华网,中国平安,招商银行,深圳能...	山东黄金,中国银行,数码科技,泰禾集团,中国平安,广州发展,招商银行,华发股份,ST中新,立...

Figure 5. First five lines of the final result

4. Discussion

4.1 Why didn't choose to use text vectorization to calculate similarity in Task1 Q1?

I mainly tried text vectorization based on the bert base chinese model. There are three reasons for not choosing this method, firstly, it has input length limitation. Its maximum length input length is 512, which does not meet the need in many cases, and if the problem is solved by segmentation or sliding window, it will bring great computational consumption. And if it is based on ORG similarity matching after NER, the length, although satisfied, will again be inching forward because of the emergence of the second reason - too long running time. The last reason is that the similarity distinction between texts is not obvious or even wrong, for example, there have been cases where the cosine similarity between ‘建行’ and ‘中国银行’ is 0.8809, while ‘建行’ and ‘建设银行’ have a cosine similarity of 0.8775.

4.2 Is there a better NER model to improve the strategy in Task1 Q1 ?

Yes. I think the spaCy-based zh_core_web_trf clearly performs better. It substantially outperforms LAC in comparisons of the same text, e.g. for the text below, LAC cannot accurately identify multiple consecutive organization names that follow. The performance of zh_core_web_trf is clearly very good. However, the slowness of its oversized model is the reason why I didn't choose it in the end. (Although it gives small and medium-sized models, the results are obviously average and the running time is not dominant.)

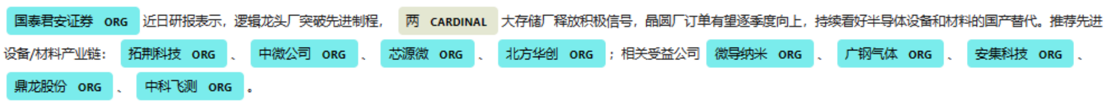

国泰君安证券 ORG 近日研报表示，逻辑龙头厂突破先进制程，两 CARDINAL 大存储厂释放积极信号，晶圆厂订单有望逐季度向上，持续看好半导体设备和材料的国产替代。推荐先进设备/材料产业链：拓荆科技 ORG 、中微公司 ORG 、芯源微 ORG 、北方华创 ORG ；相关受益公司 微导纳米 ORG 、广钢气体 ORG 、安集科技 ORG 、鼎龙股份 ORG 、中科飞测 ORG 。

Figure 6. NER recognition results for zh_core_web_trf

4.3 Why not use the table-based query method in Task2 Q4?

The main reason is the runtime, after some experimentation I found that the graph search approach would be twice as fast as a table based query for the same code flow. So I finally chose the graph search approach.

4.4 Why not pre-process the Newsdata?

In fact, I recognized many problems in the data, such as the presence of news in English. But I didn't intervene too much since I wanted to build a strategy that could be turned into a process solution where most of the problems would be solved automatically.

5. References

Github:

LAC: https://github.com/baidu/lac
spaCy: https://github.com/explosion/spaCy
SnowNLP: https://github.com/isnowfy/snownlp
py2neo: https://github.com/neo4j-contrib/py2neo
FinBERT: https://github.com/ProsusAI/finBERT
Fingpt: https://github.com/AI4Finance-Foundation/FinGPT
HiDy: https://github.com/K-Quant/HiDy
Hugging Face:

Bert-base-chinese: https://huggingface.co/bert-base-chinese
Paper:

[1] Medhat W, Hassan A, Korashy H. Sentiment analysis algorithms and applications: A survey[J]. Ain Shams engineering journal, 2014, 5(4): 1093-1113.

[2] Wang W, Xu Y, Du C, et al. Data set and evaluation of automated construction of financial knowledge graph[J]. Data Intelligence, 2021, 3(3): 418-443.