

Project Report

All my code is available on Github: <https://github.com/Kainan-Liu/DSAA5002-DataMining>

Task1.1

Considering the host memory consumption and computational complexity, I just use a **rule-based strategy and conduct a brute force search** among all the news data. Specifically, I load the A-shared list json file first and construct a name list of all the A-shared company. And then filtering based on whether any company's name appears in the news content. Therefore, I build a matrix to store the news and its corresponding company.

```
import operator
grid_search_matrix = np.ones((len(self.texts), len(name_list)), dtype=bool)
for i, text in tqdm(enumerate(self.texts), leave=False, total=len(self.texts)):
    for j, name in enumerate(name_list):
        grid_search_matrix[i, j] = operator.contains(text, name)
```

After that, we can get noise data without company names and news with company names. The filtered rate is around

$$\text{filter rate} = \frac{\# \text{ filtered news}}{\# \text{ Total news}} = \frac{473199}{1037035} = 0.4563$$

However, we can also adopt some approximate word mining algorithm and I refer to this GitHub project: https://github.com/tigerchen52/synonym_detection

With this synonym detection strategy, we can expand our candidate name list table like below:

	Company_Name1	Company_Name2	Company_Name3	Company_Name4
	建行		茅台	中国石油
	建设银行	胜利股份	贵州茅台	中石油
News1	1	0	0	0
News2	0	0	0	1
News3	0	0	0	0
News4	1	1	0	0
News5	1	1	1	0

We also adopt the rule-based strategy as before while expanding the candidate name list by synonym detection like "Baidu online encyclopedia of close synonyms"

To reduce the computational time, we use this strategy only on the noise data without company names in the A-shared name list(full name)

Task1.2

A coarse-to-fine approach

- Pretrained-Model: <https://huggingface.co/hw2942/bert-base-chinese-finetuning-financial-news-sentiment-v2>
- Train Two-layer MLP on uncertainty data

For financial sentiment analysis, It is natural to think of using a pre-trained model on unlabeled data to first perform a labeling first. Therefore, I use a Chinese Bert-based model which is pretrained on Chinese financial news. I think it is very suitable for our project.

⚡ Inference API ⓘ

🔗 Text Classification

Examples ▾

人民币兑美元中间价报7.1498，下调286点

Compute

Computation time on Intel Xeon 3rd Gen Scalable cpu: 0.036 s

Negative	0.999
Neutral	0.001
Positive	0.000

Notice that the pretrained model has a classifiers for triple classification (`classifier`): `Linear(in_features=768, out_features=3, bias=True)`, thus I provide **two options** for our tasks. One is use this classifiers explicitly and the other is: remove the logit of the "Neutral" label and apply softmax function only on Negative and Positive lable.

```
with torch.no_grad():
    for idx, inputs in loop:
        inputs = inputs.to(device = config.DEVICE)
        output = self.model(**inputs)[0]
        output = output if neutral else output[:, [0, 2]]
        outputs = torch.concat((outputs, output), dim=0)
    return outputs.detach()

def convert_label(self, outputs: torch.Tensor):
    dim = outputs.shape[1]
    logits = F.softmax(input=outputs, dim=1)
    if dim == 3:
        print("=====> label: Negative: 0; Neutral: 1; Postive: 2")
        column_indices = torch.argmax(input=logits, dim=1)
    elif dim == 2:
        print("=====> label: Negative: 0; Postive: 1")
        column_indices = torch.argmax(input=logits, dim=1)
    else:
        raise ValueError("Outputs dimension should be 2(negative/postive) or 3(negative/neutral/postive)")
```

After that, we can obtain a coarse sentiment label for every news content.

	NewsID		NewsContent	Explicit_Company	label
0	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
1	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
2	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...	外运发展,中国国航		1

The result is not bad. However, I want to get a more accurate result by relabeling the uncertainty label given by the pretrained model.

The definition of Uncertainty: After applying softmax function, if the probability for Positive and Negative is very close, we can consider it as uncertain label

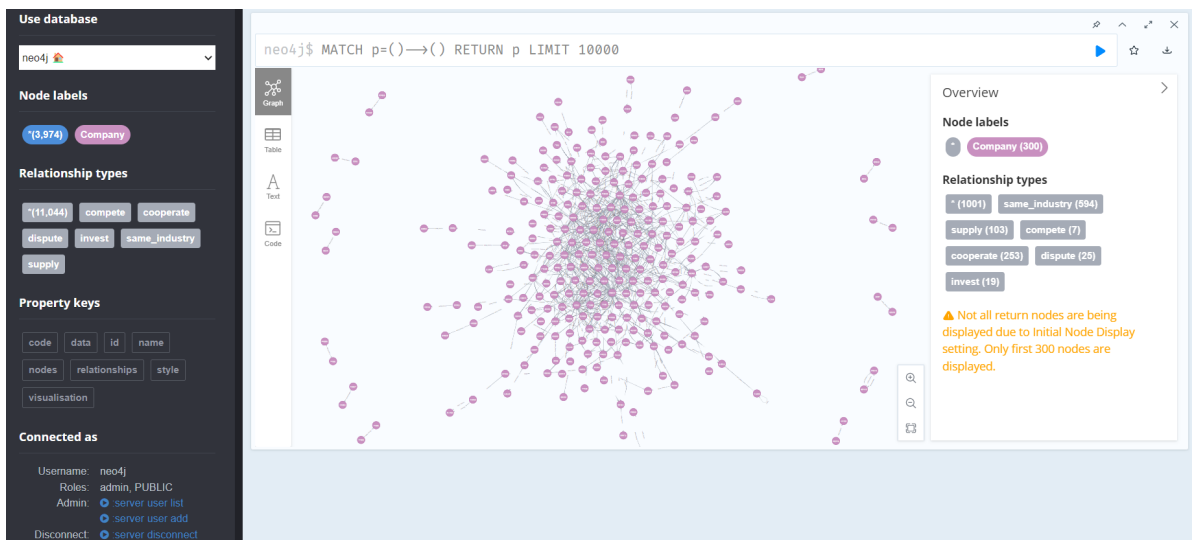
```
def select_by_certainty(self, outputs: torch.Tensor, threshold: float):
    num_sample = outputs.shape[0]
    dim = outputs.shape[1]

    logits = F.softmax(outputs, dim=1)
    unsure_row_indices = torch.where(torch.abs(logits[:, 0] - logits[:, 1])
    < threshold)[0]
    sure_row_indices = torch.where(~(torch.abs(logits[:, 0] - logits[:, 1])
    < threshold))[0]
    sure_row_indices = sure_row_indices.cpu().detach().numpy()
    unsure_row_indices = unsure_row_indices.cpu().detach().numpy()
    return sure_row_indices, unsure_row_indices
```

And then, I can train a two layer MLP in a supervised way on the data with a certain label given by the pretrained LLM and then infer on the uncertain news content

Finally, we combine the result of this two phases to obtain the final output **Task1.xlsx**

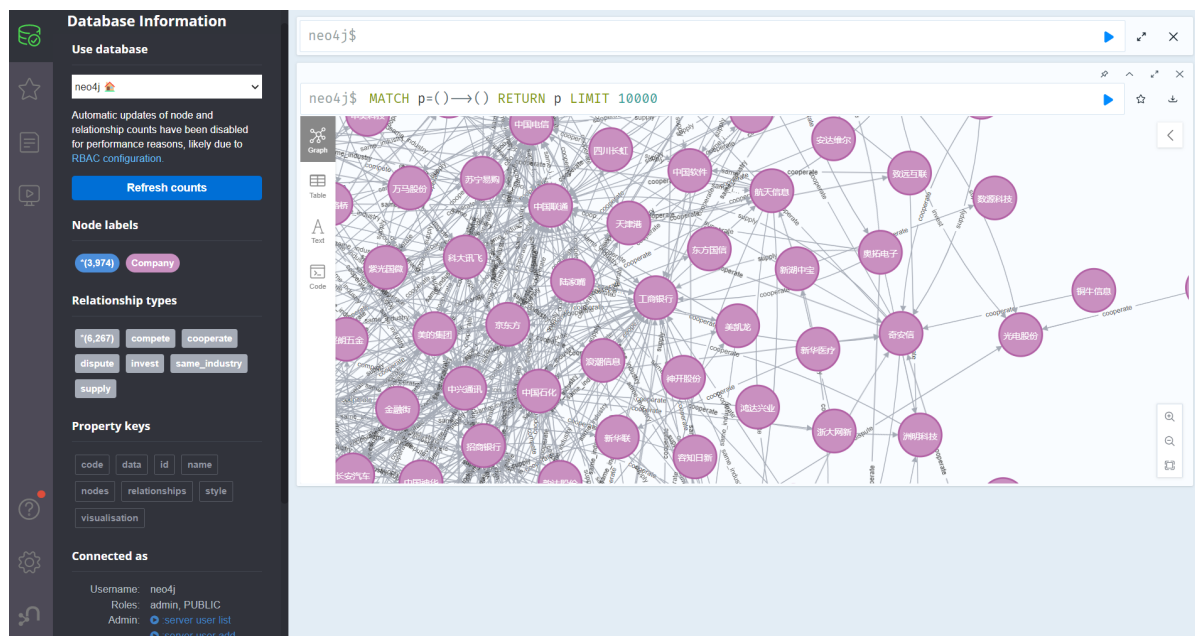
Task2.1



Py2neo is popular Python library for interacting with Neo4j databases. It provides a more abstract and higher-level API compared to the official `neo4j` driver.

Thus, I construct a knowledge graph using the API in py2neo by passing the csv files. Build the nodes with company type and other attributes like code and ID, also build the relationship using `py2neo.Relationship` between different company nodes. Again, all my code is available in <https://github.com/Kainan-Liu/DSAA5002-DataMining>

Zoom in for details:



Task2.2

Knowledge-Driven Financial Analysis: With the knowledge graph we construct in task2.1 and the rules given by the task, we can identify ALL implicit companies corresponding to each company of Explicit_Company in your own Task1.xlsx file.

Relation between company A and B	The impact of news sentiment on companies	Case 1		Case 2	
		New impact on A	New impact on B	New impact on A	New impact on B
<i>compete</i>	opposite	1	0	0	1
<i>cooperate</i>	same	1	1	0	0
<i>dispute</i>	opposite	1	0	0	1
<i>invest</i>	same	1	1	0	0
<i>same_industry</i>	same	1	1	0	0
<i>supply</i>	same	1	1	0	0

Although I adopt some parallelism strategy like Dataframe `df.apply()` to perform faster, I fail to obtain all the implicit company name within the time limit. It is very time consuming with two for loops.

```
def implicit_mining(self, *, file_path = None, num_sample):
    print("Start mining the implicit company")
    assert file_path is not None, "file_path should not None"
    if os.path.exists(file_path):
        news = pd.read_excel(file_path).iloc[:num_sample, :]
        def get_positive_implicitCompany(value):
            name_list = [name.strip() for name in value.split(',')]
            positive_nodes = []
            for company_name in name_list:
```

```

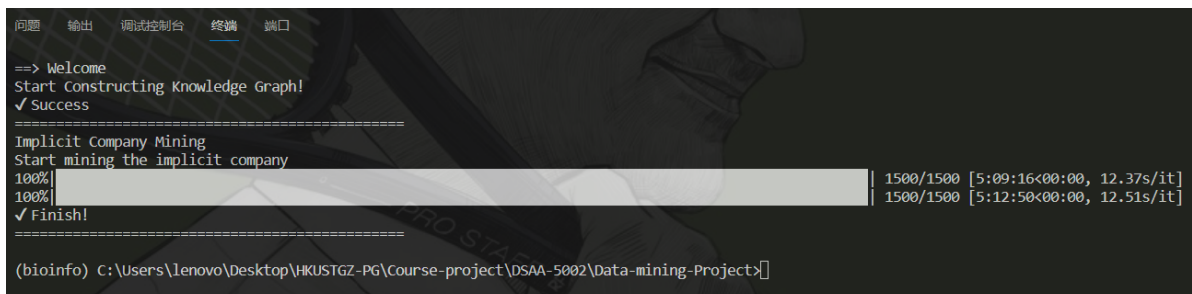
        node = self.node_matcher.match("Company",
name=company_name).first()
        relationships = list(self.relation_matcher.match(nodes=
(node, None)))
        for relationship in relationships:
            if type(relationship).__name__ in ["cooperate",
"invest", "same_industry", "supply"]:
                positive_nodes.append(relationship.end_node["name"])
            if positive_nodes == []:
                positive_nodes = None
            return positive_nodes

    def get_negative_implicitCompany(value):
        name_list = [name.strip() for name in value.split(',')]
        negative_nodes = []
        for company_name in name_list:
            node = self.node_matcher.match("Company",
name=company_name).first()
            relationships = list(self.relation_matcher.match(nodes=
(node, None)))
            for relationship in relationships:
                if type(relationship).__name__ in ["compete",
"dispute"]:
                    negative_nodes.append(relationship.end_node["name"])
            if negative_nodes == []:
                negative_nodes = None
            return negative_nodes

    tqdm.pandas()
    news["Implicit_Positive_Company"] =
news["Explicit_Company"].progress_apply(get_positive_implicitCompany)
    news["Implicit_Negative_Company"] =
news["Explicit_Company"].progress_apply(get_negative_implicitCompany)
    news.to_excel("./Data/Task2.xlsx", index=False)
else:
    raise FileNotFoundError

```

And that is the running time for 1500 samples in the datasets



```

问题  输出  调试控制台  终端  窗口
==> Welcome
Start Constructing Knowledge Graph!
✓ Success
=====
Implicit Company Mining
Start mining the implicit company
100%|████████████████████████████████████████████████████████████████████████████████| 1500/1500 [5:09:16<00:00, 12.37s/it]
100%|████████████████████████████████████████████████████████████████████████████████| 1500/1500 [5:12:50<00:00, 12.51s/it]
✓ Finish!
=====
(bioinfo) C:\Users\lenovo\Desktop\HKUSTGZ-PG\Course-project\DSAA-5002\Data-mining-Project>

```