Gradually Entering Blissful Circumstances: Report of Data Mining and Knowledge Discovery in Stock Related News

50015627 JIANG Zhuoyang

1 TASK1-Q1 FILTERING STRATEGY

1.1 Problem Analysis

The project provides a table containing over a million Chinese news articles and a structured JSON file consisting of information related to A-share companies. The first question in Task 1 requires us to filter the over a million Chinese news articles using the information related to A-share companies, retaining only the news articles that mention A-share companies.

1.1.1 *Input Data*.

The provided input data comprises two types: one is a table containing news articles, categorized under the "data processing domain," and the other is a structured JSON file containing information about A-share companies, categorized under the "information support domain." We need to extract the most effective information from the "information support domain" as a reference for data processing to ensure efficient data handling.

1.1.2 *Output Requirements*.

In terms of the result, we aim to obtain a table containing news articles that mention A-share companies, which I refer to as the "filtered result domain." The filtered table should prioritize high-quality news in two aspects: firstly, the result should preferably not contain unrelated news, i.e., "purity"; secondly, the result should ideally include all relevant news about A-share companies, i.e., "completeness." Here, I consider the effectiveness of stock news to be more crucial for subsequent analysis, hence I prioritize "purity" as the primary goal. Considering the substantial volume of data, it is crucial to ensure the "efficiency of generation" of the output.

Additionally, while filtering the news, for ease of further processing, we should also explicitly extract the relevant A-share companies mentioned in each filtered news article and store them in a new row for backup purposes. Thus, the desired filtered result domain for this problem consists of relevant news articles and the associated companies mentioned within these articles.

1.2 Iteration and Preliminary Experiments of Filtering Strategies

Based on the types of effective information in the "information support domain", I proposed two preliminary strategies, conducted experiments, and then summarized and observed patterns. I aimed to combine their strengths to construct a final strategy that performs well in terms of "purity," "completeness," and "efficiency of generation."

1.2.1 *Strategy One (Preliminary Strategy): Character Matching-Based Strategy*.

(1) Initial Thoughts: In the information support domain, stock names of A-share companies are provided, which are crucial information for stock news. Hence, using this information, traversing the list of stock names for each news article to check for any company's stock name would result in the highest "purity" in the filtered result domain. Additionally, abbreviations or aliases of companies are commonly effective information, and the other information provided in the information support domain, especially the full company names, often include all characters of the aliases. This can be used to construct rules to generate aliases, further filtering news articles without stock names but containing aliases, enhancing the "completeness" of the filtered result domain.

(2) Initial Experiment: I used both "stock names" and "rule-generated aliases" as rules to match all relevant news articles. Observing the results from these two rules: - The stock name rule ensured "purity" while keeping the stock names unchanged (i.e., not removing special characters like ST, ST*, etc.) and excluding delisted stocks. - The aliases generated from the "full name", this is a very complex rule (Due to space limitations, please refer to my code "PreliminaryExperiment-Task1-Q1" for details.) which always included common widespread words like "technology," "industry," etc. Although "completeness" slightly improved, it compromised "purity." Using this single-layered filtering method produced a basic filtered result domain containing news entries: Size_of_Experimental_domain = 657291

1.2.2 *Strategy Two (Preliminary Strategy): Similarity Matching-Based Strategy*.

(1) Initial Thoughts: In the character matching scheme, since we prioritize "purity" in results, generating aliases cautiously limits their coverage of possible real-life scenarios. Leveraging semantic vector similarity between original names and aliases could enhance the precision of alias matching from a semantic perspective. Initially, two approaches were considered: A. Brute-force sliding window: Calculate semantic similarity using a fixed sliding window for every word in each news article and each stock name. If a word's semantic similarity exceeds a threshold, include the news article in the filtered result domain. This method consumes significant time and severely violates the requirement of "generation efficiency." B. Named Entity Recognition (NER) for organizational extraction: Perform NER on each news article, extract all "organization" entities, and evaluate semantic similarity matching only on "organization" entities in each news article. This method is somewhat more efficient than the brute-force method, but the time complexity remains high.

(2) Initial Experiment: Using BERT tokenizer [1] to generate semantic vectors and employing the brute-force sliding window for similarity screening, I noticed two unacceptable points: Firstly, it consumed excessive time, taking approximately a month on my laptop, completely failing to meet the "generation efficiency" requirement. Secondly, the precision of the filtered results was low due to matching many irrelevant words since many companies (e.g., "XUEREN(Snowman)") design their names using everyday words.

Although this issue can be mitigated by using the NER-based extraction, it also incurs unacceptable time consumption.

1.2.3 *Strategy Three (Final Strategy)*.

Throughout the strategy iteration and initial experiments, I attempted to combine the advantages of both approaches to propose the final strategy.

(1) Observation on "Fullname Domain Highly Covers Alias Domain": Firstly, concerning the name information in the news, I discovered a crucial point: news containing aliases generally also include full company names. Therefore, the number of news articles that can only be filtered using aliases is relatively low.

(2) Reversed Approach - Clever Use of Similarity Strategy: Secondly, regarding the similarity strategy, it's not necessary to employ it in the vast "data processing domain." Instead, using the similarity strategy in the much smaller "information support domain" could be effective and meet the output's "generation efficiency" requirement. Hence, by utilizing self-defined alias generation rules in the "information support domain" to create some aliases, determining the similarity between these aliases and full names using the similarity strategy, generating accurate aliases, and then using these aliases to construct character matching rules, we can ensure a certain level of "purity" while expanding the "completeness" of the output.

1.3 Execution of the Final Strategy

Based on the initial experimental conclusions and the three requirements for output: "purity," "completeness," and "efficiency of generation," the final strategy was determined and executed. The process and results are presented as follows:

1.3.1 *Hierarchical Filtering for Different Domains*.

(1) Construction of the "Tight Domain" using Stock Full Names: Considering the preliminary experimental results of the character matching strategy and observations regarding the "full name domain highly covering the alias domain," it's noted that using stock full names can create a "tight domain" with high "purity" and relatively high "completeness." This tight domain serves as the primary output domain named the "training domain," while the remaining news articles form the "second-level data processing domain." Ultimately, the "tight domain" contains:

$$\text{Size\_of\_Training\_domain} = 475242$$

(2) Clever Use of Alias Generation and Similarity Strategy: I improved the rule from the preliminary experiments to a better rule (Also quite complex. Due to space limitations, please refer to my code "Task1-Q1" for details.) and used it to generate aliases. I further reviewed the quality of aliases by categorizing the quantity of aliases with different word counts. As two-word aliases often contained common words, special attention was given to them, manually filtering out common words as shown in Figure 1.

['爱慕', '仙鹤', '综艺', '雪人', '大洲', '起步', '锡业', '光电', '地矿', '胜利']

Figure 1: An example of common words

Following this, as each company alias is generated from its original name, I directly calculated the cosine similarity of these aliases and their corresponding original names using the semantic vectors (generated by the "bert-base-chinese" model's tokenizer), determining whether it exceeded a threshold (here, I chose 0.7 as the threshold). This process resulted in 1568 usable aliases. This step ensured a certain level of "purity" and expanded the "completeness" of the output.

Using these aliases to filter news from the "second-level data processing domain" formed a new output known as the "supplementary domain." Although the "purity" of this output is not as high as the "tight domain," it is assured by the alias similarity threshold and also expands a certain level of "completeness."

Eventually, the "supplementary domain" contains:

$$\text{Size\_of\_Supplementary\_domain} = 48984$$

(3) Summarizing Final Entries and Extracting Companies to Obtain the Required `Task1.xlsx`. Constructing a trie for "stock names" in the "training domain," finding all relevant companies, and adding their corresponding "stock names" to the "Explicit_Company" column. Constructing a trie for "aliases" in the "supplementary domain," finding all relevant companies' aliases, using the "stock name - alias" dictionary to find corresponding "stock names," and adding them to the "Explicit_Company" column. (The "stock names" in the "Explicit_Company" column are separated by ",") Merging the results of the "training domain" and "supplementary domain" to obtain the final "application domain," which contains:

$$\text{Size\_of\_Application\_domain} = 475242 + 48984 = 524226$$

1.3.2 *Significant Methods to Address Massive Data Volume during Execution*.

Previous work has somewhat guaranteed the requirements of "purity" and "completeness" in the output. However, due to the massive volume in the "data processing domain," various techniques were employed in different stages of the output process to ensure "generation efficiency." Two specific techniques were implemented:

- **Multi-threaded programming**: Utilizing multi-threading for parallel or concurrent processing in the two-level matching and explicit company extraction significantly improved processing efficiency.
- **Trie search**: Constructing a trie for company "stock names" or "aliases" during explicit company extraction drastically enhanced search and extraction efficiency. Compared to using multiple loops for search and extraction, constructing the trie reduced processing time from the "several days" level to the "several tens of minutes" level.

1.4 Filter Rate Summary

The filter rate shows below:

(1) Total number of news articles in the "data processing domain" .

$$\text{Size\_of\_Total\_domain} = 1037035$$

(2) Filter Rate of Experimental Single-layered Filtering Strategy:

$$\text{Filter\_rate\_experiment} = \frac{\text{Size\_of\_Experimental\_domain}}{\text{Size\_of\_Total\_domain}} = \frac{657291}{1037035} \approx 0.6338$$

(3) Filter Rate of Final Two-level Filtering Strategy:

$$\text{Filter\_rate\_final} = \frac{\text{Size\_of\_Application\_domain}}{\text{Size\_of\_Total\_domain}} = \frac{524226}{1037035} \approx 0.5055$$

2 TASK1-Q2 SENTIMENT ANALYSIS METHODS

2.1 Problem Analysis and Initial Approach

In the first problem, we acquired A-share company-related news that meets good levels of "purity" and "completeness." Now, the goal is to perform sentiment analysis on these news articles to explore their impact on companies and even the stock market.

2.1.1 *Task Requirements and Qualification*.

The sentiment analysis task in this project is modeled as a binary classification machine learning problem. Approaches for analyzing sentiment tendencies in news roughly fall into two directions: one involves "directly using" existing high-quality sentiment analysis models for all news, while the other involves "self-training" sentiment analysis models specific to news related to the stock market.

I believe this task requires balancing proficiency in sentiment analysis and expertise in analyzing stock news tendencies. Existing sentiment analysis models often excel in "proficiency" but lack the required "expertise" for stock market news.

Regarding the second direction, as long as we balance "proficiency" and "expertise" during data annotation, we can train excellent "self-training models." Hence, I've primarily chosen the second direction and introduced the "proficiency" offered by existing models during the data annotation phase. This involves using the confidence information provided by the sentiment analysis classification quality of existing models to annotate data during the "self-training" process. This annotated data, combined with specialized annotations for news, forms the training set for the sentiment analysis model tailored for news.

Thus, the output requirements for this task are similar to general binary classification machine learning problems, primarily necessitating a model with "accuracy" and "generalizability." Here, balancing proficiency in sentiment analysis and expertise in analyzing tendencies related to stock news leads to optimal strategies for "data annotation" and "model selection" to ensure "accuracy" and "generalizability." Additionally, efficiency is crucial for each step in this task.

2.1.2 *Input Form*.

Initially, I explored categorizing the input as "single-company news" and "multi-company news," conducting different annotation and model training for each category. Company information and news were simultaneously treated as input to make the model's tendency analysis more practical. However, considering that this task only requires sentiment analysis of news without immediate consideration of their impact on the contained companies, this approach remained exploratory. Eventually, only the news input mode was chosen for this task.

2.2 Iteration of Data Annotation Strategy and Preliminary Experiments

To select a better data annotation strategy, aiming for "accuracy" in output, I believe data annotation should consider both "maturity in sentiment analysis capabilities" and "professionalism in stock news analysis":

2.2.1 *Strategy One (Preliminary Strategy) - Limitations of Independent Manual Data Annotation*.

Initially, I attempted to manually label 1000 samples. However, due to my limited knowledge of stock-related matters, working independently, and slow reading speed, I couldn't ensure accuracy or efficiency in the labeling process. Thus, I sought other strategies.

2.2.2 *Strategy Two(Preliminary Strategy) - Leveraging LLM Chatbot for "Professionalism"*.

After analyzing the limitations of independent manual data annotation, I envisioned using an efficient large language model (LLM) chatbot, which possesses comprehensive knowledge and efficient working capabilities, for data annotation.

(1) Method Description: LLM chatbots operate based on conversational systems, so I used a suitable prompt to simulate the thought process of a data annotator. Using a fixed prompt question specifically tailored for sentiment analysis, each news piece was combined with this question as a prompt target. Then, using an API in batches, these prompts were inputted into the LLM chatbot to receive feedback for annotating the sentiment of stock news.

(2) Initial Experiment and Pros & Cons Analysis: After testing with the efficient LLM chatbot ERNIE-Bot-turbo from Wenxin Yiyan [4], I summarized the advantages and disadvantages of this approach: - Pros: Strong relevance to stock news, providing comprehensive explanations. It assures a level of accuracy from a "professionalism" standpoint. - Cons: It's not a specialized sentiment analysis model. It lacks confidence information, which is essential for sentiment analysis tasks ("proficiency"). It also runs inefficiently and is relatively expensive, which dissuaded me from using this chatbot for a full-scale analysis of all data.

2.2.3 *Strategy Three (Preliminary Strategy) - Leveraging a Mature Sentiment Analysis Model for "Proficiency"*.

Addressing the proficiency limitation of LLM chatbots, I further explored using a mature sentiment analysis model's data annotation capabilities:

(1) Method Description: By directly employing Baidu Wenxin Yiyan's "Sentiment Analysis" API, I could utilize the outstanding Chinese sentiment analysis model known as Senta. This model takes Chinese strings as input and outputs a dictionary containing "confidence", "positive_prob", "negative_prob" and "sentiment" values. With this model, we can annotate a large volume of training samples.

(2) Pros & Cons Analysis: After testing with some news samples using this method, I highlighted the pros and cons of this approach: - Pros: A specialized sentiment analysis model providing confidence information. It assures a level of accuracy from a "proficiency" standpoint. - Cons: Lacks strong relevance to stock news, lacks "professionalism."

2.2.4 *Strategy Four (Final Strategy) - Integration of LLM Chatbot and Mature Sentiment Analysis Model*.

To combine the "professionalism" from the LLM chatbot and the "proficiency" from the mature sentiment analysis model, I annotated a hundred thousand samples using both methods. The key challenge became "how to consolidate the annotations from both pipelines."

I initially considered utilizing cleanlab's crowd-sourced data annotation management functionality [2]. However, cleanlab's methods (like the CROWDLAB algorithm) mainly cater to scenarios with many annotators, each with sparse annotations. As I used only two annotators for dense annotations, I directly defined the rule: When both annotations match or one result is missing, I adopt the available annotation. If the mature sentiment analysis model, Senta, provides a confidence score higher than the threshold (0.65) while the results differ, I adopt Senta's result; otherwise, I adopt ERNIE's result. For samples where both annotations are missing due to various reasons (like network disconnection, illegal queries), I manually input these into ChatGPT to get results, finally consolidating the data annotation results.

2.3 Model Strategy Iteration

During the self-training of the model, from the "training domain" data acquired in Question 1, I selected 100,000 data points to construct a training set. There were two model options available: a smaller BiLSTM and a larger model, BERT.

2.3.1 *Strategy One (Preliminary Strategy): BERT Model*.

Based on the theory of large models BERT's exceptional performance of fine-tuning (as shown in Figure 2) [3], my initial thought was to introduce the pre-trained large model `bert-base-chinese` for transfer learning and fine-tuning on our binary classification task.

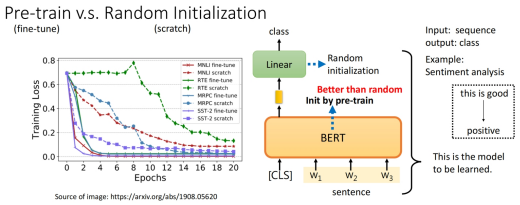


Figure 2: The exceptional performance theory of fine-tuning

***(1) Fine-tuning Experiment with the Large Model* `BERT`:.**

a **Exploration One: Fine-tuning on Entire Training Samples:**

Due to limited computational resources, fine-tuning BERT was time-consuming. Using 90% of the 100,000 samples for fine-tuning was not feasible on my CPU. In the Colab T4GPU environment, fine-tuning for more than 2 epochs would result in disconnection. Even in this setting, using the remaining 10% of test samples achieved an accuracy of 0.7769.

b **Exploration Two: Fine-tuning with Fewer Samples of High Confidence:**

Reducing the number of epochs and training samples for efficiency, I selected 5000 positive and 5000 negative samples from those where both Senta and ERNIE provided identical annotations with the highest confidence. This created a 10,000 sample training dataset for training on 90% and testing on 10%. After training for 10 epochs, the test accuracy reached 0.9960. However, when I validated with 10,000 randomly selected samples from the original 100,000, the accuracy dropped to 0.5677. This suggests that high-confidence samples severely weaken the "generalization" of the model, indicating it's not a good strategy.

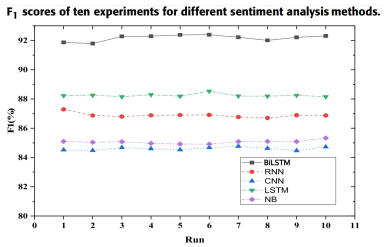
c **Exploration Three: Fine-tuning with Random Fewer Samples:**

Training on 10,000 randomly chosen samples, using 90% for training and 10% for testing, for 10 epochs achieved only an accuracy of 0.7983.

(2) Conclusion: Due to my limited computational resources, finding a balance between the number of fine-tuning iterations and the size of the training samples posed a dilemma, preventing me from obtaining a highly effective model.

2.3.2 *Strategy Two (Final Strategy): Smaller BiLSTM Model*.

Since Bi-LSTM is highly suitable for sentiment analysis in text classification [5] (as depicted in Figure 3), it has shown excellent results in text analysis. Given that the task itself isn't overly complex, BiLSTM can yield good results without necessitating the use of the BERT model. With my limited computational resources, the BiLSTM model's training epochs and training sample sizes can simultaneously meet requirements, allowing for an effective model.



G. Xu *et al.*: Sentiment Analysis of Comment Texts Based on BiLSTM

Figure 3: The superiority of BiLSTM in semantic segmentation tasks

2.4 Execution of Final Strategy:

2.4.1 *Construction of Training Set:*

Randomly extracted 100,000 samples from the most accurate filtered A-share-related news result obtained earlier, termed the "training domain."

2.4.2 *Secondary Data Annotation*.

Utilized both the LLM chatbot ERNIE-Bot-turbo and the mature Wenxin Yiyan sentiment analysis model (referred to as "senta," derived from Baidu's well-known sentiment analysis model) APIs to annotate the 100,000 samples.

(1) Annotation of Data via LLM Chatbot ERNIE-Bot-turbo: Employed the official API call instances, defining a prompt similar to Figure 4 to receive feedback. Assigned a label of 1 for "positive news" and 0 for "negative news" based on the received feedback. Utilized multithreading and addressed connection interruptions by implementing wait-recovery processes during batch annotation to ensure efficiency and stability.

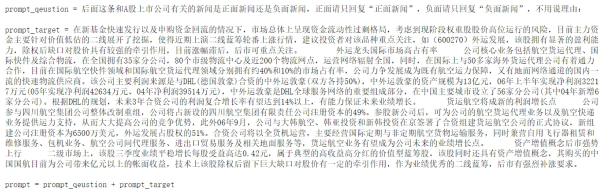


Figure 4: Prompt Engineering of LLM

(2) Annotation of Data via WenxinYiyan Sentiment Analysis API:. Utilized the official API call instances. Following the pattern as shown in Figure 5, obtained the final sentiment label based on the model's output, recording key supplementary information such as "confidence." Similar to the LLM chatbot, ensured efficiency and stability using multithreading and wait-recovery processes during batch annotation.

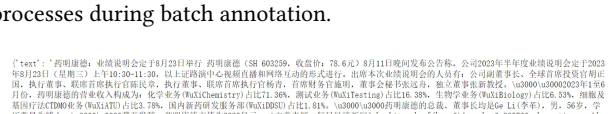


Figure 5: Output pattern of WenxinYiyan Sentiment Analysis API

(3) Integration of Data Annotation Results: Employed the rule to integrate the data annotation results. As illustrated in Figure 6, for the majority of news, the annotations from both sources align. However, for cases where annotations differ, considering the higher confidence in the big model's annotations, it's vital to use its specialized expertise and explanatory ability for logical judgments.

```
Final_Sentiment
1    78831
0    21169
Name: count, dtype: int64
Annotation_State
consensus                  75581
difference_trust_ERNIE     14129
difference_trust_Senta     10143
ERNIE_missing_trust_Senta     74
Senta_missing_trust_ERNIE     72
manually_ask_ChatGPT           1
```

Figure 6: Data Annotation Results

2.4.3 *Model Training*.

Utilized the "bert-base-chinese" model's tokenizer to create 640-dimensional word vectors for each news item. Followed the settings shown in Figure 7 to train the model. The final Bi-LSTM model achieved a test accuracy of 0.8882 after the 48 epochs and at last 0.8799 after 64 epochs on a 10,000-sample test set which shows a good performance.

```
encoding = self.tokenizer.encode_plus(
    text,
    add_special_tokens=True,
    max_length=640,
    padding='max_length',
    truncation=True,
    return_tensors='pt'
)
```

```
input_dim = tokenizer.vocab_size
hidden_dim = 128
output_dim = 2
dropout = 0.1

batch_size = 64
lr = 2e-4
epochs = 64
```

Figure 7: Model configuration

2.4.4 *Model Application*.

Applied the BiLSTM model to conduct sentiment analysis scoring on the "application domain" dataset obtained from the first question. The distribution of the two categories of sentiment-oriented news is depicted in Figure 8.

```
label
1    416302
0    107924
Name: count, dtype: int64
```

Figure 8: Distribution of sentiment-oriented news

3 TASK2-Q3

4 NODE GENERATION:

Generate a node for each company in *hidy.nodes.company.csv*.

4.1 Relationship Generation

a **Bi-Directional Relationship Generation:**
Utilize several CSV files corresponding to *cooperate*, *dispute*, *same_industry*, and *compete* to create bidirectional relationships within the graph.

b **Uni-Directional Relationship Generation:**
Utilize CSV files corresponding to *supply* and *invest*. to create single directional relationships within the graph.

4.2 Knowledge Graph Visualization

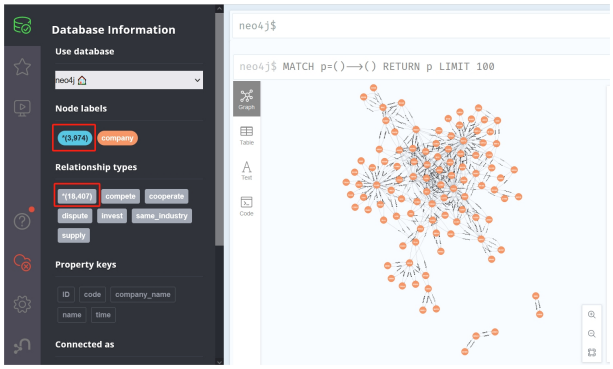


Figure 9: Knowledge Graph Visualization

REFERENCES

[1] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, and Ziqing Yang. 2021. Pre-training with whole word masking for chinese bert. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), 3504–3514.

[2] Hui Wen Goh, Ulyana Tkachenko, Jonas Mueller, and Cleanlab Cleanlab Cleanlab. 2022. CROWDLAB: Supervised learning to infer consensus labels and quality scores for data with multiple annotators. *arXiv preprint arXiv:2210.06812* (2022).

[3] Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2019. Visualizing and understanding the effectiveness of BERT. *arXiv preprint arXiv:1908.05620* (2019).

[4] Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, et al. 2021. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2107.02137* (2021).

[5] Guixian Xu, Yueting Meng, Xiaoyu Qiu, Ziheng Yu, and Xu Wu. 2019. Sentiment analysis of comment texts based on BiLSTM. *Ieee Access* 7 (2019), 51522–51532.