Method of Domain Specific Code Generation Based on Knowledge Graph for Quantitative Trading

Jianshui Bi, Hongming Cai, Bo Zhou, Lihong Jiang

▶ **To cite this version:**

Jianshui Bi, Hongming Cai, Bo Zhou, Lihong Jiang. Method of Domain Specific Code Generation Based on Knowledge Graph for Quantitative Trading. 11th International Conference on Research and Practical Issues of Enterprise Information Systems (CONFENIS), Oct 2017, Shanghai, China. pp.21-33, 10.1007/978-3-319-94845-4_3 . hal-01888627

HAL Id: hal-01888627

https://inria.hal.science/hal-01888627

Submitted on 5 Oct 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Method of Domain Specific Code Generation Based on Knowledge Graph for Quantitative Trading

Jianshui Bi[1], Hongming Cai[1], Bo Zhou[2], and Lihong Jiang[1]

[1] Shanghai Jiao Tong University, Shanghai, China
[2] Jiangsu Hoperun Software Co., Ltd, Jiangsu, China
[1] {bijianshui, hmcai, jianglh}@sjtu.edu.cn
[2] zhou_bo1@hoperun.com

Abstract. Quantitative methods have been adopted by more and more individual investors for investment activities. Many third party platforms have been developed to help users complete the process of backtesting, which fills the gap between the trading strategy code and the trading strategy model. However, using a quantitative platform for backtesting has a high threshold for users who do not have programming experience. There is still a huge gap between the description and the code of trading strategy. Code generation allows developers to focus more on business related design and implementation, thereby increasing the efficiency of software development. The import of domain knowledge can improve the accuracy of requirement parsing to improve the quality of constructed code model. The general knowledge base is often incomplete in terms of domain specific terms and relationships, and the construction of domain knowledge graphs requires more domain related data. In this paper, encyclopedia data and the financial report data are used to extract domain terms and relations. And then a domain knowledge graph for quantitative trading is constructed to realize the automatic generation of quantitative trading strategy code.

Keywords: code generation, knowledge graph, quantitative trading.

1 Introduction

Quantitative trading takes the advantage of mathematical models instead of artificial subjective judgments, thus avoiding the adverse effects of human emotions in financial products trading. In addition to institutional investors, more and more individual investors have begun to invest through quantitative methods. The key step in establishing a quantitative trading model is to validate the effectiveness of the proposed model. Historical data are used to simulate the model performance in history. This process is often referred to the backtesting of a quantitative trading strategy.

There are many third party backtesting platform providers at present, which provides the historical market data and the transaction simulation program package. Users can write code of trading strategy on the platform, which will be processed based on historical market data using the trading simulated engine provided by the platform. Then an evaluation report is generated to tell users the performance of the strategy in

history, including the rate of return, the sharp ratio, etc., which is supposed to improve the strategy for users. These platforms enable users not need to obtain historical transaction data in advance, nor to deal with the logic of analog transactions, and to calculate the transaction results, greatly reducing the threshold for users to develop quantitative trading strategies.

However, the existing quantitative trading backtesting platform usually requires users to follow certain rules defined by the platform. Strategies are supposed to be written to code in a specific programming language. Users without engineering background usually have a high learning cost to be skilled in a programming language. And because different platforms are implemented based on different frameworks, the code of trading strategies between different platforms is often difficult to migration.

Code generation is to generate computer programs by some mechanisms, thus allowing software engineers to program at a higher level of abstraction, which can improve the quality of the code. Code generated quality only depends on the code template, code model and the data file. Developers can focus more on business related design and implementation, which improves software development efficiency. Code generation can also reduce the difficulty for code migration in different frameworks. It takes the business logic into the language independent model, and code for different frameworks can be generated by providing code templates for different frames based on the same business logic model.

Code generation is usually based on different forms of requirements descriptions, and requirements descriptions are highly relevant to application domains. Domain terms need to be identified in natural language requirement descriptions. Domain knowledge graphs, as a collection of domain terms and relationships, can be used for text requirement descriptions identification. This paper uses the domain specific data of open knowledge base and open data as the data source to build the knowledge graph in the field of financial analysis. By the analysis of the user's strategy description text based on the domain knowledge, the strategy code in backtesting platform can be generated. Therefore, users can focus more on the strategy, instead of the policy of written code about the specific platform. Users can develop quantitative trading strategies to guide investment activities without programming skills, which reduces the threshold for users to use the quantitative trading backtesting platform.

The paper is presented as follows: Section 2 describes the related work; Section 3 provides an overview of framework; Section 4 describes the proposed method for the domain knowledge graph construction for quantitative trading; Section 5 presents the method and the example of generate code for a specific backtesting platform; and Section 6 gives the conclusion of this paper.

2 Related Work

Code generation usually takes different forms of requirement description as input. Through the analysis and modeling of requirement description, combining code template or code generation rule, the program source code is generated. Requirement description is usually based on the specific application domains, including a large

number of terms in the field of text description. The identification of domain terms is a prerequisite for accurate identification of requirements in the domain text, as well the relationship between domain terms and program execution logic. As a collection of domain terms and relationships, domain knowledge graphs can play an important role in the identification of text requirement descriptions.

Alkhader Y et al. [1] used the natural language description of the requirements document as input, automatically generate the corresponding UML class diagram design. After processing with natural language related technology to generate requirements documents described in XML format, it is necessary for requirements engineers to use domain knowledge to manually eliminate redundancy and solve similarity problems. Popescu D et al. [2] transformed the constraint grammar representation requirements specification statements into object oriented analysis model based on domain terms, to help identify the manual review of ambiguity and inconsistency. The grammar and part of speech rules are used to recognize the domain terms.

Bolloju N et al. [3] introduced the knowledge based model quality and model ontology to evaluate the quality of object model, so that the model constructed has better semantic quality. Li G et al. [4] proposed an engineering process that enables domain ontologies to guide the requirements elicitation process. Kong D et al. [5] using the dom4j analytical framework and the Velocity template engine, automatic generation of database definition language, database manipulation language and the specific operation page code for information management system. Wei Y et al. [6] used predefined code templates to generate project code that meets the specific J2EE MVC framework based on FreeMarker. Lopata A et al. [7] proposed that import enterprise model and enterprise meta model as knowledge data source in the model driven architecture development process.

As a collection of domain terms and relationships, domain knowledge graphs have important applications in the fields of natural language processing and other fields. The extraction of domain terms and relationships is the core work of domain knowledge graph construction. Hua B et al. [8] used a rule-based approach to identify sentences containing technical terms from academic literature. Technical terms were extracted based on combining the rules and the vocabulary lists. Song P et al. [9] used dependency parsing and automatic annotation of semantic roles to formulate extraction rules and weights, and extract knowledge units automatically from the term definition sentences.

The goal of relation extraction is to extract the fact between entities, and the relation extraction problem is usually abstracted into a two classification problem, that is, to determine whether the selected two entities have the relation. Perera R et al. [10] put forward the characteristics of words, semantic features and other dimensions for the application of relation extraction. Chen C et al. [11] proposed to use iterative methods to extract patterns of specific relationships based on predefined entity relationships, and then further extract relationships.

In this section, we present a brief overview of importing domain knowledge in code generation process, as well the term extraction and relation extraction in knowledge graph construction.

3 Overview of Framework

Fig. 1 gives an overview of domain knowledge graph construction framework, while Fig. 2 gives the framework of strategy code generation based on the domain knowledge graph. The input in knowledge graph construction phase is the Wikipedia and Baidu encyclopedia data as the data source for term extraction and relation extraction, while XBRL (Extensible Business Reporting Language) format report data as the data source for relation extraction, and the output is the domain knowledge graph for quantitative trading, including the domain terms and the relations between domain terms. The input in strategy code generation phase is the strategy description written in natural language following some rules, and the output is the strategy code for a specific quantitative trading backtesting platform which is determined by the *code template* and the *comparison table* which is shown in Fig. 2.

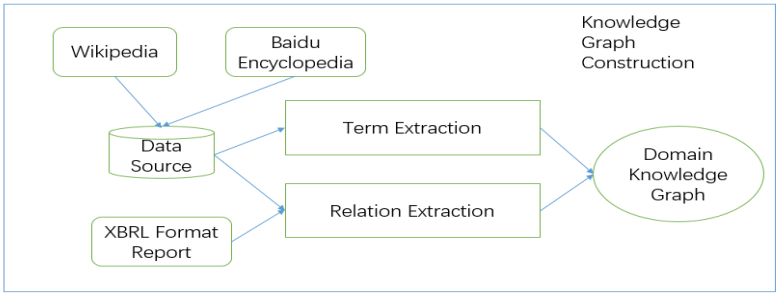


Fig. 1. Framework of domain knowledge graph construction.

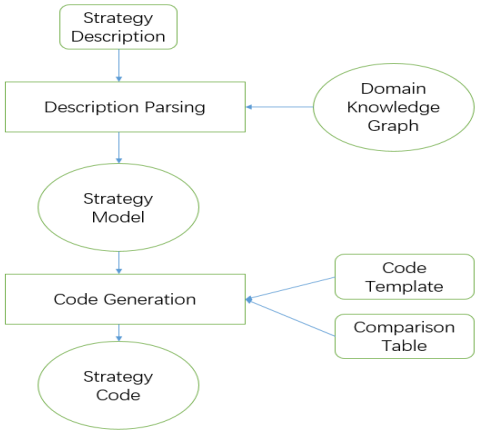


Fig. 2. Framework of strategy code generation.

The method of domain specific code generation based on domain knowledge graph for quantitative trading mainly has the following steps:

1. Term Extraction: In this step, the domain terms of financial analysis for quantitative trading need to be extraction for later relation extraction. Encyclopedia data are used as the data source to extract domain terms by defined rules according to the features different from common words.
2. Relation Extraction: In this step, the relation between domain terms needs to be extraction for domain knowledge graph construction. We use some domain specific text in open data in addition to encyclopedia data to establish the relation between domain terms extracted in above step.
3. Description Parsing: In this step, the quantitative trading strategy written in natural language need to be parsed. The domain terms are identified based on domain knowledge graph. And the domain knowledge in the domain knowledge graph is used to construct the strategy platform independent model.
4. Code Generation: In this step, the quantitative strategy code can be executed in a specific quantitative trading backtesting platform will be generated based on the strategy model and the code template related to the specific backtesting platform.

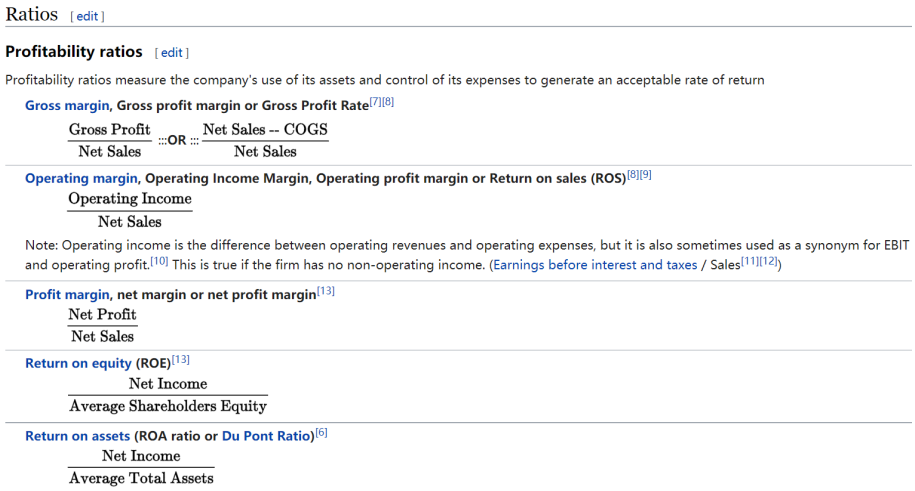
In this section, we give a brief overview of the framework we proposed in this paper. We use encyclopedia data and domain specific data in open data to extract domain terms and the relation between domain terms. And a domain knowledge graph for quantitative trading is constructed to provide domain knowledge in domain specific code generation.

4 Domain knowledge graph construction

In this section, we describe the method of constructing a domain knowledge graph in financial analysis for quantitative trading, including the domain terms exaction and the term relation extraction.

4.1 Domain Terms Extraction

To construct a domain knowledge graph, we need to extract the terms in the domain firstly. This paper builds a knowledge graph that applies to financial analysis, so it is necessary to extract the terms in the field of financial analysis. There are some entries to summarize the relevant terms in the encyclopedic data and domain terms can be extracted from them. For example, *financial ratios* entry in Wikipedia and Baidu encyclopedia includes *inventory turnover rate* and *total asset turnover ratio*, and *financial analysis entry* in Baidu encyclopedia contains *price earnings ratio*, *price to book ratio*, etc. An example of financial ratio entry in Wikipedia is shown in Fig. 3.


Ratios [edit]

Profitability ratios [edit]

Profitability ratios measure the company's use of its assets and control of its expenses to generate an acceptable rate of return

Gross margin, Gross profit margin or Gross Profit Rate[7][8]

$$\frac{\text{Gross Profit}}{\text{Net Sales}} \text{ :::OR::: } \frac{\text{Net Sales -- COGS}}{\text{Net Sales}}$$

Operating margin, Operating Income Margin, Operating profit margin or Return on sales (ROS)[8][9]

$$\frac{\text{Operating Income}}{\text{Net Sales}}$$

Note: Operating income is the difference between operating revenues and operating expenses, but it is also sometimes used as a synonym for EBIT and operating profit.[10] This is true if the firm has no non-operating income. (Earnings before interest and taxes / Sales[11][12])

Profit margin, net margin or net profit margin[13]

$$\frac{\text{Net Profit}}{\text{Net Sales}}$$

Return on equity (ROE)[13]

$$\frac{\text{Net Income}}{\text{Average Shareholders Equity}}$$

Return on assets (ROA ratio or Du Pont Ratio)[6]

$$\frac{\text{Net Income}}{\text{Average Total Assets}}$$

Fig. 3. An example of financial ratio entry in Wikipedia.

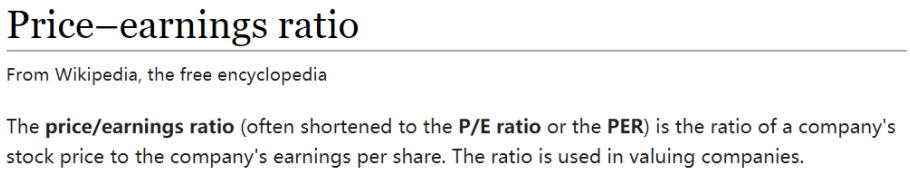
These domain terms in summary entries usually have common characteristics. For example, in the field of financial analysis, domain terms usually associated with formula. At the same time the related domain terms also has some domain independent common characteristics. For example, if the word contains hyperlinks to the other entry, the word is more likely to be a term. In this paper, these features are used to define rules for extracting domain terms of financial analysis from relevant summary entries. And These domain terms will be used for subsequent relation extraction and domain knowledge graph construction.

4.2 Term Relation Extraction

In the previous section, the extraction of domain specific terms has been completed. This section describes the method of extracting the relationships between domain terms based on the domain terms extracted from the previous section. The data source of relation extraction mainly includes two parts, one is the entry pages in the encyclopedia, and the other is the other open data related to the field as a supplement.

For the term entry page in the encyclopedia, we use pattern matching method to extract relations between domain terms. An example of same as relation in *price earnings ratio* entry in Wikipedia is shown in Fig. 4. For example, *price earnings ratio* entry has the following statement in Baidu encyclopedia, also known as *market price earnings ratio*, *pe ratio*, and *PER*. The *price earnings ratio*, *market price earnings ratio*, *pe ratio*, and *PER* are the same meaning in the field of financial analysis can be represented by three tuples are as follows:

- < price earnings ratio, same_as, market price earnings ratio >
- < price earnings ratio, same_as, pe ratio >
- < price earnings ratio, same_as, PER >


Price–earnings ratio

From Wikipedia, the free encyclopedia

The **price/earnings ratio** (often shortened to the **P/E ratio** or the **PER**) is the ratio of a company's stock price to the company's earnings per share. The ratio is used in valuing companies.

Fig. 4. An example of same as relation in price earnings ratio entry in Wikipedia.

For the extraction of relation between domain terms, the formula of field terms can be obtained by matching the patterns of specific computational symbols. For example, with the term *quick ratio*, the Baidu encyclopedia describes the entry as equation (1).

$$\text{Quick Ratio} = \text{Quick Assets} / \text{Current Liabilities} \tag{1}$$

The segmentation of the formula to a symbol of operation, we can get the relationship in three domain terms about *quick ratio*, *quick assets* and *current liabilities*, and the *quick ratio* is equal to the *quick assets* divided by *current liabilities*. This relationship can be represented by a set of tuples:

- < quick ratio, f1_dividend, quick assets >
- < quick ratio, f1_divider, current liabilities >

The *f1* in the above tuples means that the tuples are number 1 group statements for the *quick ratio* term. In this relation, *quick assets* is as the dividend in *quick ratio* calculation, and the *current liabilities* is as the divider in the *quick ratio* calculation. The relationship can be expressed as *quick ratio* equal to the *quick assets* divided by the *current liabilities*.

For the domain terms extracted from the above mentioned method, the relations are extracted from the corresponding entries in the encyclopedia. The relationship between domain terms can be obtained to establish a domain specific knowledge graph.

However, the encyclopedia data usually depends on the manual editing, which means the cover of the specific domain terms may not be very complete. This paper uses other open data related to this domain, in order to enhance the degree of coverage of the domain terms. XBRL (Extensible Business Reporting Language) format of standard financial report has been more and more accepted by the exchange and the listed company. The calculation link library including in the XBRL format financial report can be used as a supplement for encyclopedia of data to domain term relation extraction.

XML format is used in extensible business reporting language format of financial report to describe the calculation relation. We defined rules to transform XML format description to our three tuple format in domain knowledge graph.

The following is an example of an extensible business reporting language format of financial report:

- < arcrole = http://www.xbrl.org/2003/arcrole/summation-item from = “gross profit” to = “sales revenue” order = “1” weight = “1” />

- < arcrole = http://www.xbrl.org/2003/arcrole/summation-item from = “gross profit” to = “cost of sales” order = “2” weight = “-1” />

The http://www.xbrl.org/2003/arcrole/summation-item means the items described in the code are the total sum of the relationship. In this relationship, the *gross profit* is the summary item, while the *sales revenue* and the *cost of sales* are the add items. The order attribute means that in this relationship, *sales revenue* is the first and *cost of sales* is the second. The weight attribute means that in this relationship, the weight of *sales revenue* is 1 and the weight of *cost of sales* is -1. The code fragment represents the relationship between *gross profit*, *sales revenue* and *cost of sales*, follows the equation (2).

$$\text{Gross Profit} = \text{Sales Revenue} - \text{Cost of Sales} \tag{2}$$

Converting the above relation into a three tuple format expresses as follows:

- < gross profit, f1_minuend, sales revenue >
- < gross profit, f1_subtrahend, cost of sales >

Fig. 5 shows a part of constructed domain knowledge graph for quantitative trading. The relation between domain terms in constructed domain knowledge graph can be represented by tuples. The *net operating profit rate* is equal to the *net profit* divided by *operating income* as an example, the following tuples can be used to be the representation of the relationship.

- < net operating profit rate, f1_dividend, net profit >
- < net operating profit rate, f1_divider, operating income >

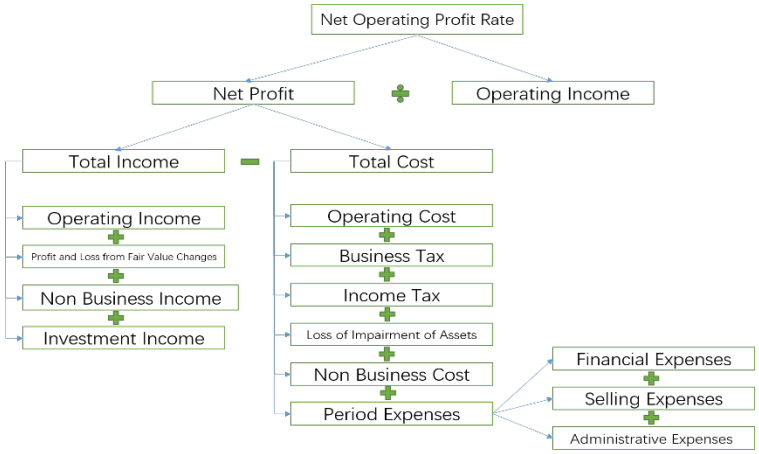


Fig. 5. A part of constructed domain knowledge graph for quantitative trading.

Two tuples above mean that the *net operating profit rate* can be obtained by the calculation of the *net profit* and the *operating income*, the *net profit* for the dividend, the

operating income for the divider, namely the *net operating profit rate* is equal to the *operating income* divided by the *net profit*. And *f1* means it is the first groups related to the *operating net profit rate*. The same entity tends to have multiple sets of relational representations, such as *price earnings ratio*, which can be calculated either by *stock prices* and *earnings per share*, or by *market capitalization* and *net profit*.

For the *total income* equals to the summary of *operating income*, *profit and loss from fair value changes*, *non-business income* and *investment income*, the following four tuples can be expressed:

- < total income, f1_addend, operating income >
- < total income, f1_addend, profit and loss from fair value changes >
- < total income, f1_addend, non-business income >
- < total income, f1_addend, investment income >

This section presents the method of constructing domain knowledge graph for quantitative trading using the encyclopedia data and the domain specific data, including the method of term extraction and the method of relation extraction.

5 Domain specific code generation

In this section, we describe the method and the example of generating code for a specific backtesting platform, including the strategy description parsing and the backtesting platform code generation.

5.1 Strategy Description Parsing

The user's strategy description can be parsed based the domain terms in the domain knowledge graph constructed above. The strategy description needs to include the start date and the end date of the backtesting, trading target, and the execution logic of the strategy. A description of the user strategy logic condition needs to be followed by certain priority order rules.

In the following strategy description as an example, the trading target is ticker 000540, if the net operating profit rate is more than 20% and the current stock price is less than 10 yuan, or the moving average of the stock price in 5 days is more than the moving average in 10 days, open (buy); otherwise close (sell). The start date of backtesting is January 1, 2016, and the end date of backtesting is December 31, 2016.

The generated strategy model is illustrated as Fig. 6.

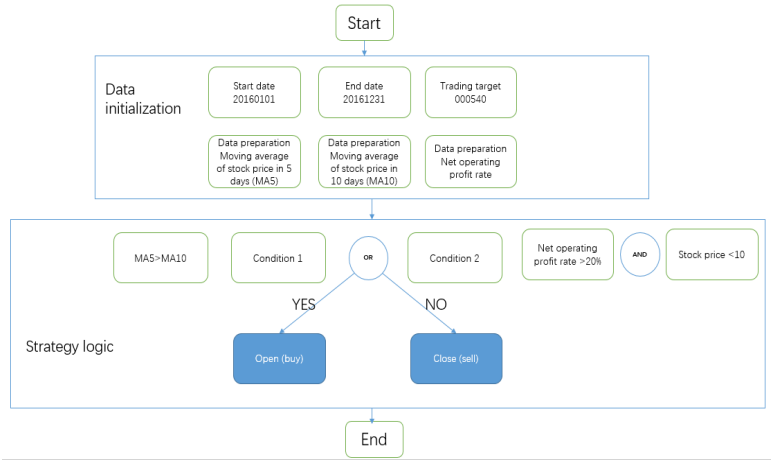


Fig. 6. An example of generated strategy model.

In the phase of data initialization, the backtesting start time (20160101), end time (20161231), and trading target (000540) are initialized. And the moving average of the stock price in 5 days, the moving average in 10 days and the *net operating profit rate* which required for the *net profit* and *operating income* (or further the *total cost*, *total income*, and *operating income*) need to be extracted in this phase as well. Then the data will be used in the strategy simulation stage to determine the conditions of strategy by calculation.

And in the phase of the strategy simulation, by modeling strategy logic, based on the extracted data in the initialization phase, the target backtesting platform uses historical market data to calculate whether the stock price or other data meets the strategy opening or closing conditions. Between the start date and the end date, the backtesting platform will simulate the trading according to the strategy conditions by calling the logic of the strategy. Then the backtesting platform will generate a report about the strategy performance in the period determined by the start date and the end date by processing the simulated trading record. The report usually includes indicators such as return rate, sharp ratio, etc., which provides a reference for the users to improve the strategy.

5.2 Backtesting Platform Code Generation

The section above describes the method to transform the user requirement descriptions of quantitative trading strategies to platform independent model based on the constructed domain knowledge graph for generating backtesting platform strategy code. This section will describe the method of generating strategy code which can be executed by the backtesting platform based on the platform independent model and the platform specific code template.

For specific quantitative trading backtesting platform, code templates are written using the FreeMarker template language defined by the FreeMarker template engine.

The code to specify the trading target, start date, end date of backtesting and extract the data in the initialization phase should be included in the templates. And the templates should also include the code to generate the logic code of the strategy according to the strategy logic model. The platform should also give a comparison table to define the transform relation between the term in domain knowledge graph and the data field name in the platform. The FreeMarker engine is used to generate code that can be executed by a specific quantitative trading backtesting platform, based on the strategy model which is constructed by parsing strategy descriptions.

```
def initialize(self):
    ticker_id = '000540'
    from_time = time.strptime("2016-01-01 09:15:00", "%Y-%m-%d %H:%M:%S")
    to_time = time.strptime("2016-12-31 15:15:00", "%Y-%m-%d %H:%M:%S")
    self.add_data('Tick', ticker_id, 'data', ['TimeNum', 'LastPX'],
                  time.mktime(from_time), time.mktime(to_time))
    self.add_transform('MovingAverage',['data'], ['LastPX'], 600)
    self.add_transform('MovingAverage',['data'], ['LastPX'], 2400)
```

Fig. 7. An example of generated data initialization code.

Fig. 7 shows the generated code of data initialization, including the trading target, start date, end date and the moving average of the stock price. Fig. 8 presents the generated strategy code and Fig. 9 gives an example of trading strategy backtesting report generated by a backtesting platform.

```
def handle_data(self, data):
    ticker_id = '000540'
    now_time = data['data.TimeNum']
    price = data['data.LastPX']
    short_ma = data['data.LastPX.movinga_average600']
    long_ma = data['data.LastPX.moving_average2400']

    if (self.compare == -1 and short_ma > long_ma):
        if (self.ordered == False):
            self.order(ticker_id, True, True, 'LimitOrder', price + 0.02, 100, now_time)
            print now_time
            self.trade_is_buy = 1
            self.ordered = True
        else:
            self.order(ticker_id, True, False, 'LimitOrder', price - 0.02, 100, now_time)
            print now_time
            self.trade_is_buy = 0
            self.ordered = False
    elif (self.compare == 1 and short_ma < long_ma):
        if (self.ordered == False):
            self.order(ticker_id, False, True, 'LimitOrder', price - 0.02, 100, now_time)
            print now_time
            self.trade_is_buy = -1
            self.ordered = True
        else:
            self.order(ticker_id, False, False, 'LimitOrder', price + 0.02, 100, now_time)
            print now_time
            self.trade_is_buy = 0
            self.ordered = False
    if (short_ma > long_ma):
        self.compare = 1
    elif (short_ma < long_ma):
        self.compare = -1
```

Fig. 8. An example of generated strategy logic code.

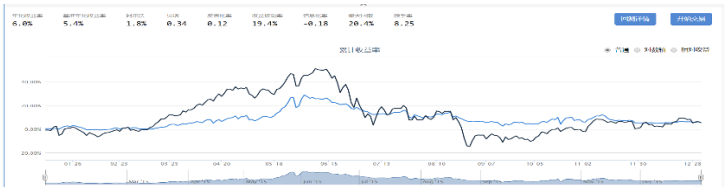


Fig. 9. An example of backtesting platform generated report.

This section we describe the method and the example of generating code for a specific backtesting platform. And the method of strategy description parsing and the backtesting platform code generation are presented in detail.

6 Conclusion

The third party backtesting platform providers make it is possible for individual investors to develop quantitative trading strategies without establishing a complete backtesting framework. However, it is difficult for users to migrate the strategy code because of the difference between these platforms. And there is also a huge gap between the description and the code of trading strategy especially for investors without programming skills.

Code generation is usually used to enhance the efficiency of application development. We use code generation related technologies in this paper to generated quantitative strategy code. In previous research, domain knowledge is rarely used in the code generation process. This paper uses the relevant data of open knowledge base and open data as the data source, and constructs a domain knowledge graph in the field of financial analysis to analyze user's strategy description text. Then quantitative trading strategy code in the backtesting platform is generated to reduce the threshold for users to use the backtesting platform. It allows users to focus more on research and development of quantitative trading strategies, while the development of the strategy can be easier migrated among different platforms. This paper has constructed the domain knowledge graph of financial analysis applying in Chinese listed companies. While the construction of knowledge graph depends on the financial statements of listed companies to disclose China criteria, because of different national accounting rules and legal differences, which may not be applied to the process of financial analysis of foreign companies directly.

In the future research, we will apply domain knowledge graph in more fields, such as intelligent agriculture, where knowledge graph could be used for agricultural product tracking.

Acknowledgement

This research is supported by Key R&D Project of Zhejiang Province under Grand No.2017C02036.

References

1. Alkhader Y, Hudaib A, Hammo B. Experimenting With Extracting Software Requirements Using NLP Approach[C]// International Conference on Information and Automation. IEEE Xplore, 2007:349-354.
2. Popescu D, Rugaber S, Medvidovic N, et al. Reducing Ambiguities in Requirements Specifications Via Automatically Created Object-Oriented Models[C]// Innovations for Re-

quirement Analysis. From Stakeholders' Needs To Formal Designs, Monterey Workshop 2007, Monterey, Ca, Usa, September 10-13, 2007. Revised Selected Papers. DBLP, 2007:103-124.
3. Bolloju N, Sugumaran V. A knowledge-based object modeling advisor for developing quality object models[J]. Expert Systems With Applications, 2012, 39(3): 2893-2906.
4. Li G, Jin Z, Xu Y, et al. An Engineerable Ontology Based Approach for Requirements Elicitation in Process Centered Problem Domain[C]// Knowledge Science, Engineering and Management -, International Conference, KSEM 2011, Irvine, Ca, Usa, December 12-14, 2011. Proceedings. DBLP, 2011:208-220.
5. Kong D, Luo F, Lin W, et al. RESEARCH ON A VELOCITY-BASED AUTOMATIC CODE GENERATION TECHNOLOGY[J]. Computer Applications & Software, 2014.
6. Wei Y, Management S O, University T. Backstage Management System Code Generator Based on J2EE and Maven[J]. Computer & Modernization, 2014.
7. Lopata A, Ambraziunas M. Knowledge-Based MDA Approach[C]// Business Information Systems Workshops - BIS 2011 International Workshops and BPSC International Conference, Poznań, Poland, June 15-17, 2011. Revised Papers. DBLP, 2011:160-165.
8. Hua B. Extracting Information Method Term from Chinese Academic Literature[J]. New Technology of Library & Information Service, 2013.
9. Song P, Lu Q, Liu N. A New Method for Knowledge Unit Automatic Extraction Using Definitions of Terms[J]. Journal of Intelligence, 2014.
10. Perera R, Nand P. A Multi-strategy Approach for Lexicalizing Linked Open Data[M]// Computational Linguistics and Intelligent Text Processing. Springer International Publishing, 2015:348-363.
11. Chen C, He L, Lin X. REV: extracting entity relations from world wide web[C]// International Conference on Ubiquitous Information Management and Communication. ACM, 2012:1-5.