2022 年美国数模赛电子科技大学模拟赛

承诺书

我们仔细阅读了数学建模竞赛的竞赛规则.

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子 邮 件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其 他公开的资料(包括网上查到的资料)，必须按照规定的参考文献的表述方式在正 文引用处和参考文献中明确列出。我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违 反竞赛规则的行为，我们将受到严肃处理。

我们的题目编号是（填写：A 或者 B）：B 题

我们报名**队伍**号是：H131

参赛队员姓名学号（**中文填写打印并签名**）：

1. 陶砚青+2020040401013

2. 杨皓凯+2020010801013

3. 石荃+2020140901013

指导教师或指导教师组负责人（有的话填写）：

是否愿意参加 2022 年美国赛（是，否）：是

日期：2022 年 11 月 28 日

For office use only	Team Control Number	For office use only
T1	H131	F1
T2		F2
T3	Problem Chosen	F3
T4	B	F4

2022 Mathematical Contest in Modeling (MCM) Summary Sheet

Summary.

Knowledge graph is an important tool for people to search and classify knowledge. It can solve comprehensive problems for people and provide a systematic cognitive structure. Therefore, this paper constructs a knowledge graph system including communication, reconnaissance, detection and interference domains.

In the first question, this article obtains tags, attributes, and pairwise relationships in the fields of communication, reconnaissance, detection, and interference from the attached data, and realizes the visualization of knowledge graph tags and levels through the Neo4j graph database supported by JDK11. Then use the indicators in the information retrieval model to fuse multiple indicators into one value, and evaluate the label design according to the size of the value.

In the second question, this article proposes two principles for determining the relationship between labels to measure the similarity of labels. Then use the adjacency matrix notation to transform the problem into the shortest path problem. Next, the breadth-first search algorithm (BFS) is used to find the path, which quantifies the connection between the two tags in the knowledge graph.

In the third question, this article sets the rank value of the node through the geometric sequence model, and calculates the distance between the nodes and the correlation coefficient from this, and then uses this as the standard to recommend the label. Then use crawler search to crawl in a relatively small range to get the knowledge item set. In the evaluation, this article mainly considers the comprehensive evaluation using feedback data including coverage rate, failure rate, scatter degree and Gini coefficient.

In the fourth question, this article mainly considers the use of convolutional neural network algorithm to train and self-correct the parameters of the recommendation algorithm, including the rank value, the generalized value Dis between two points, etc. The training data is the feedback data in the third question. When a new knowledge item set appears to be added to the knowledge graph, firstly, the conditional random field (CRF) algorithm is used to extract the keywords and attributes of the knowledge, and the relationship between the knowledge item sets is determined through calculation of similarity and Chinese grammar analysis. Then the Prim algorithm is used to establish the new edge.

Keywords: Knowledge Graph, Information Retrieval,

Convolutional Neural Network, BFS, Neo4j

Contents

1 Introduction

1.1 Background

In an era of information explosion, we often need knowledge of many disciplines to solve a problem. How to acquire knowledge in various fields has become a big challenge. If we can obtain information through a knowledge graph that contains the correlation between knowledge, we can greatly improve our decision-making efficiency.

Under such circumstances, knowledge graph emerged.

1.2 Literature Review

There is a confusing situation, "an unforeseen race of developing new task specific graph systems, query languages and data models, such as property graphs, key-value, wide column, resource description framework"[**1**].

However, if the data grow and change in time, its necessary to improve the way to identify patterns and semantics. For this reason we will present some related new work[**2**].

Sen Hu, Lei Zou and Jeffrey Xu Yu[1] put forward method based on graphing is proposed to answer natural language questions, and their sentence structure depends on trees has more fault tolerance rate[**3**].

Another topic in research is the technical optimization of the knowledge graph and database. P. Zhao and J. Han address the graph query problem on large networks by decomposing shortest paths around vertex neighbors as basic indexing unit.

At last, another way of optimization should be mentioned. In the context of GIS graph databases [3] try to optimize the heuristic for shortest paths. At the same times we must notice that their way will use increasing time if solve with complex and large graphs.

1.3 Restatement of the Problem

We are required to establish a knowledge graph system. The knowledge graph system should be able to provide the related items as close as possible by inputting the keywords. In addition, the system can decide which classifies the given information belongs to.

The following is our restatement of each small question:

1) Design the reasonable label for items of the knowledge graph system and complete at least two kind of labels.
   - Establish a mathematical model to evaluate the performance of our knowledge graph system.
2) Give the rules to connect different knowledge items by using obtained labels.
3) Give the approach to generated the set of knowledge items under the restricted conditions.

- Establish a mathematical model to evaluate the performance of our knowledge graph system.

Give the evaluation criterion for the performance.

4) Give a mathematical model or algorithm so that the knowledge graph system is able to continue to learn to improve the content or structure.

5) Give us a knowledge graph implemented with Neo4j and Python based on the given data.

2 Assumptions

1) The data from knowledge base will not change very fast over time.

2) The aspects covered in this knowledge graph only include communication, reconnaissance, detection and interference.

3) Single arrow and double arrow can appear at the same time in this knowledge graph.

4) There will be local correlation between different dimensions of features.

5) Words that often apprear simultaneously in the same batch of documents of information are semanticallt related.

3 Notations

Symbols	Description	Unit
q	parent label's rank value	-
$rank_0$	firt-level label's rank value	-
$Dis(i,j)$	narrowly defined distance between i and j	-
$D(i,j)$	generalized distance between i and j	-
P	correct rate	-
R	recall rate	-
F	weighted average of P and R	-
G	a simple undirected graph	-
V	vertex set of G	-
E	edge set of G	-
W	adjacency matrix	-
k	weight of each edge	-
λ	similarity	-
f (time $_j$)	upward convex function	-
$d(u_0, v_0)$	smallest weighted sum between two specified vertices	-
$O(*)$	cost of *	-
S	growth scale	-
cov	coverage	-
G_i	Gini coefficient	-
FR	failure rate	-
D	dispersion degree	-
FR	failure rate	-

4 Model

4.1 Overview - Knowledge Tree

Knowledge tree model:

In view of the hierarchical relationship of different knowledge in the knowledge graph, we thought of using graph theory and the tree structure in the data structure to express the relationship of the tags in the knowledge graph. The nodes in the tree represent the knowledge tags at all levels, and the edges connecting the nodes in the tree represent the relationship between the knowledge tags and the attributes of the tags. It is worth noting that the edges we use are directed edges, that

is, the direction is from the parent node to the child node.

By analyzing the data given in the title, we found that its arrangement is a linear relationship table obtained by traversing a knowledge tree (as shown in the figure) in a certain order. We first bring the data in it into the knowledge tree structure we build to get a knowledge label map. We found that there is a special situation in the picture-loop, for example: microwave communication and wireless communication, as shown in the figure:

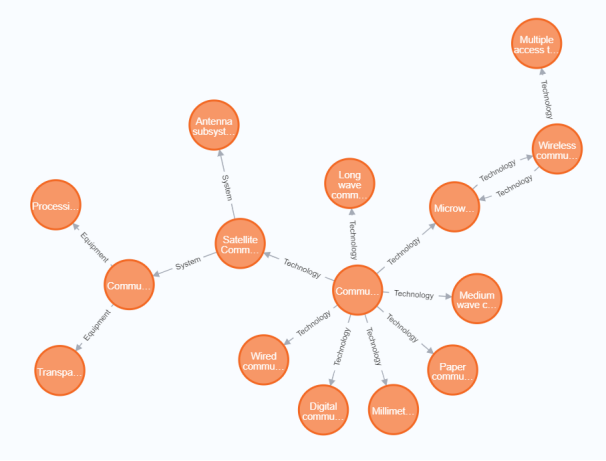

Figure 1: Part of knowledge tree

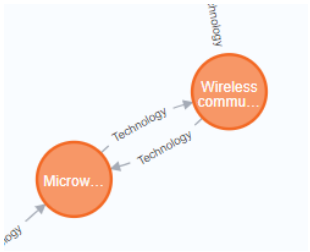

Figure 2: Ring in the tree

This shows that our graph structure cannot be simply regarded as a tree, so we decided to use a combination of solid and dashed lines to describe the knowledge graph in more detail. That is, the solid line represents the strong correlation in the relationship between father and son, and the dashed line represents the poorer correlation in the relationship between brothers and even further than the solid line.

After doing this work, we assign a rank value to each node, which will be related to the distance and the correlation of the label node.

- **Definition 1**: (label size and level description-rank value) Labels of the same level have the same rank value, and the child label is a multiple of the parent labels rank value q, where q is a fixed value and $q < 1$. We also need to set the highest value separately. The rank value of the first-level label is $rank_0$, which is similar to the domino definition method in mathematical induction. The recurrence relation of $rank$ value is as follows:

$$\mathrm{rank}_{n+1} = q \cdot \mathrm{rank} \tag{1}$$

Next, we use a defined rank value to describe the distance between nodes, and we divide it into a narrow distance and a generalized distance.

- **Definition 2**: (Narrowly defined distance) The narrowly defined distance between any two nodes is the minimum weighted distance from one node to another. The calculated value of the distance between each connected node is the absolute value of the rank difference.

$$\mathrm{Dis}(i,j) = |\mathrm{rank}_i - \mathrm{rank}_j| \tag{2}$$

(Generalized distance) The generalized distance refers to the weighted average of the narrow distance and the distance of all other nodes that contain the same key. The mathematical formula is as follows:

$$D(i,j) = \frac{\mathrm{Dis}(\dot{i},j) + \mathrm{rank}_i \cdot \frac{1}{\sum_{x=1}^{m_i} \frac{1}{\mathrm{Dis}(x,j)}} + \mathrm{rank}_j \cdot \frac{1}{\sum_{y=1}^{m_j} \frac{1}{\mathrm{Dis}(y,i)}}}{1 + \mathrm{rank_i} + \mathrm{rank_j}} \tag{3}$$

In this way, we have completed the construction of the knowledge tree model.

4.2 Sub-model 1: The Evaluation of Information Retrievals

The establishment of the knowledge graph system is to achieve this goal: input keywords and other information, make similar recommendations, or determine which category it belongs to. In the process of similarly recommending related items, the goal is to predict the knowledge content that users may want to continue to search for next. We need to ensure that the recommendations given are indeed the part of all tags that are most closely related to the input keywords. Therefore, **each tag should be assigned an evaluation index to measure its closeness to the input keywords**. When judging which category a given information belongs to, suppose we build the knowledge graph system into a tree, then we should start from the root node of the tree and go all the way to the leaf nodes, and output each node in the path in order.

After understanding the purpose of establishing the knowledge graph system, we discuss the evaluation index of the knowledge graph system's performance. When we determine the theme of a tree (child nodes, e.g. communication), a set of tags can be extracted from our knowledge graph system. This process should satisfy some constraints, including **readability**, **relevance**, and **conciseness**.

Currently, there are **two ways** to evaluate the knowledge graph system:

- ***Manual evaluation***. It is evaluated by experts in the field. The advantage of this method is that it is **highly maneuverable**, but the disadvantage is that there are differences in cognition between people, and words may have ambiguities.

- ***Comparative evaluation***. For the knowledge graph system under a certain topic that we designed by ourselves, we can also input this topic on the **authoritative academic website**, and extract the most likely "sub-nodes" synthesized by the big data of academic papers under the topic. We consider the knowledge **searched by both authoritative academic websites and the knowledge graph we established** is correct. The correct number is divided by the total amount of knowledge we have obtained as the **evaluation value**. But the scope of such consideration is not complete. We learn from the evaluation indicators in the information retrieval model and can propose more rigorous evaluation methods.

But in fact, four situations may arise when evaluating multiple tags:

The correct label was extracted; the wrong (irrelevant) label was extracted; the correct label was not extracted; the wrong (irrelevant) label was not extracted.

$$P = \frac{\text{correct extracted}}{\text{extracted}} \tag{4}$$

$$R = \frac{\text{correct extracted}}{\text{total}} \tag{5}$$

When P and R are closer to 1, the accuracy is higher.

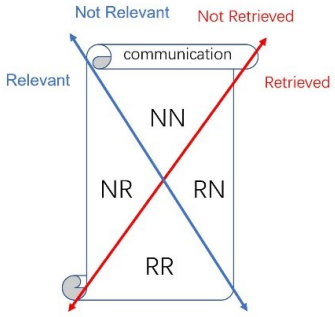

Figure 3: Relevant and retrieved

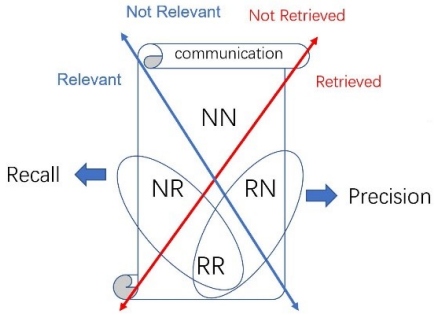

Figure 4: Recall and precision

An example: When we query Q, assuming that there should be 100 related documents, a system returns 200 documents, 80 of which are really related documents, then $R = 80/100 = 0.8$, $P = 80/200 = 0.4$. This means that the recall rate is relatively high, but the correct rate is relatively low.

The indicators P and R measure two different aspects of the system. However, in specific implementation, often only one indicator can be used, so the two indicators can be combined.

$$F = \frac{(\partial^2 + 1)\, P \times R}{\partial^2 (P + R)} \tag{6}$$

F is the weighted average of the recall rate R and the correct rate P.

In order to facilitate the implementation of the optimization algorithm in the following text, we first retain this parameter. For a set of label data $d_1, d_2, ..., d_n$(n is the total number), the system's evaluation score F,$F \in (0, 1)$. The closer F is to 1, the better the performance of the system.

4.3 Sub-model 2: Connection between Labels

This model will use the obtained tags to **give rules for connecting different knowledge items**.We use **two principles** to summarize the rules for connecting two labels.

The first principle is that we **regard the knowledge map as a family genetic map**, and the two tags with **closer** relatives should be given **higher priority** when recommending. The nature of relative distance is the shortest path problem between two specified vertices in the graph. In the following discussion, we assume that $G = (V, E)$ is a simple undirected graph, the vertex set $V = v1, v2, ..., vn$, the edge set $E = e1, e2, ..., en$, Remember $|V| = n$, $|E| = m$.

In order to implement the algorithm on the computer, a data structure can be used to describe the graph and the network, and *the adjacency matrix notation* can be used. Compared with the sparse matrix notation, if the number of edges m is much smaller than n, it is easy to cause a waste of space, but the number of edges in the knowledge graph we construct is large, so it is reasonable to use the adjacency matrix notation. Let the adjacency matrix be written as $W = \{W_{ij}\}_{n\times n}$,and k represents the weight of each edge. When G is a weighted graph, $w_{ij} = k_{ij}$ (when v_i and v_j have edges) $w_{ij} = 0$ (when v_i and v_j have no edges). For the knowledge graph we designed, the farther away from the root node, the farther away from the leaf node When it is close, the weight of the edge should decrease accordingly.

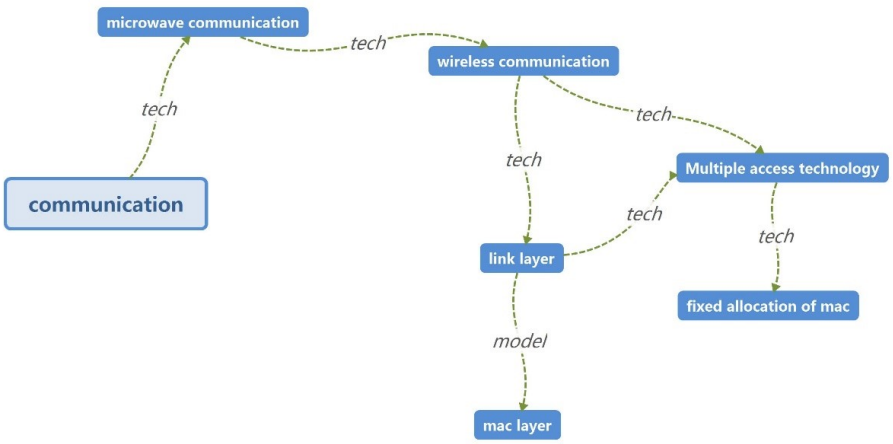

Figure 5: Part of the knowledge graph

Lets use the label *fixed allocation of max* as an example. In our map, the output of its Cypher search function is:

Obviously, *from wireless communication* to *Multiple access technology*, it may be directly connected through *technology*, or transfer through two *technology* connections of *link layer*. When querying output, we pursue simplicity, so we choose a shorter path, that is, we choose to directly

```
MATCH (n:Field)
match(m:Field{name:'Communication'})-[r]-
(n:Field{name:'Microwave communication'})-[rr]-
(a:Field{name:'Wireless communication'})-[rrr]-
(b:Field{name:'Multiple access technology'})-[rrrr]-
(c:Field{name:'Fixed allocation of MAC'})
return m,r,n,a,b,c,rr,rrr,rrrr
```

Figure 6: Output of its Cypher search function

carry out the technology connection.

When discussing the relationship between *mac layer* and *fixed allocation of MAC*, we choose:

mac layer-<link layer-> Multiple access technology-> fixed allocation of MAC

,instead a longer path going through *wireless communication*.

The second principle is that when evaluating the similarity between two tags (a, b), the higher their word **similarity**, the higher the **priority** of each tag when it is recommended. The similarity of $(SCPC, SCPC)$ is 1, and$(SVDprecoding, GMDprecoding)$ also has the keyword *precoding*, which is not exactly the same, so the similarity is set to λ_i

There are some frequently used words in the industry. If such words appear in both labels, the persuasiveness of improving the similarity measurement of these two words does not seem to be that strong. Assuming that the number of times a word $word_j$ appears in the entire knowledge graph is $time_j$, we can use an **upward convex function** $f(time_j)$ to correct it, and the function here can be selected by the user. The revised similarity is divided into $\lambda_i \times f\left(\text{ time }_j\right)$.

When comprehensively assessing the connection between the two labels, we should unify the two principles. Assuming that the path with the smallest weighted sum between two specified vertices u_0, v_0 in the weighted graph G is $d(u_0, v_0)$, in the known initial and target states, we use the *breadth first search (BFS)* algorithm, which is a traversal strategy for connected graphs.

Assuming that the graph has V vertices and E edges, the breadth-first search algorithm needs to search for V nodes, so the cost here is $O(V)$. In the search process, the length of the queue needs to be increased according to the edge, so $O(E)$ needs to be consumed here. In general, the efficiency is about $O(V + E)$.

4.4 Sub-model 3: Knowledge Search and Performance Evaluation Model

From the definition and calculation of the knowledge tree model in the model overview, we can come to this model because we use a **geometric sequence** when setting the rank value. Obviously, a new sequence formed by the difference between the last term of the geometric sequence and the previous term. The increase is quadratic, so that the distance parameter approximately **obeys the quadratic function distribution**, so the distance value we set (no matter the distance in the narrow sense or the distance in the general sense) increases with the increase of the label level, the *growth scale* can be approximated as $S\left(x^2\right)$.

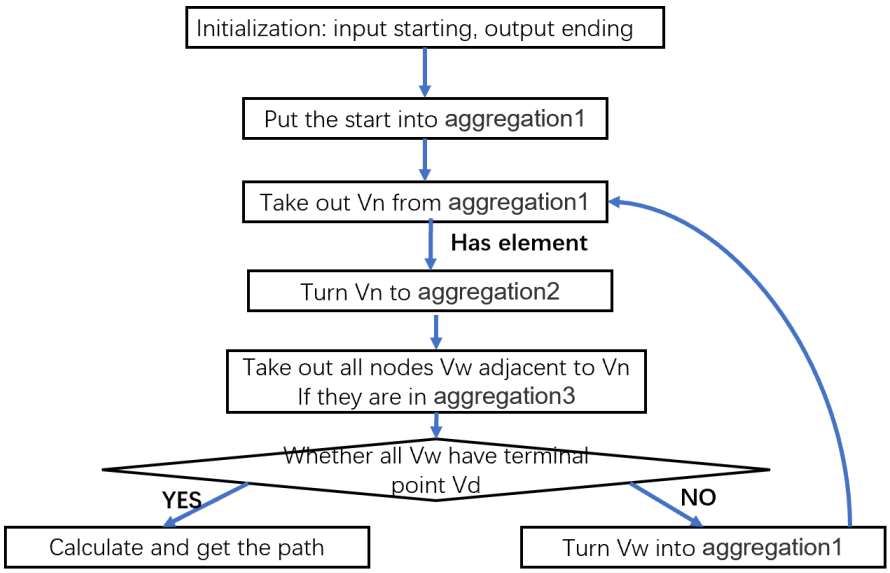

Figure 7: The breath first search(BFS)

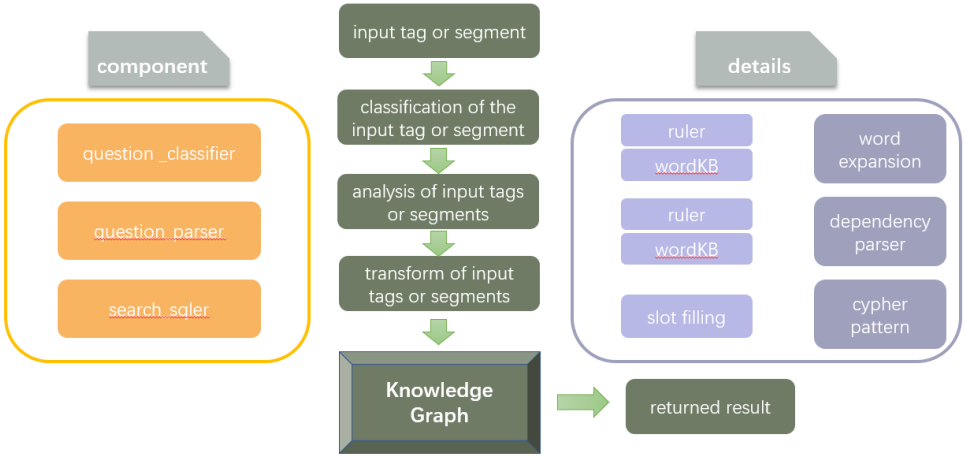

Figure 8: Similarity recommendation framework based on knowledge graph

Next, we make **clear rules on the search method and scope**:

The words searched by the user may appear in different levels. If the keywords are reorganized to form a complete sub-item (which may be very detailed), then we **recommend this sub-item first**, and then we recommend this sub-item. Secondly, we recommend **a subtree rooted at this subitem** (only 4 levels are recommended downwards), and then **start recommending 2 levels upwards** (the weighted distance can be used to calculate this range) (if there is no upward or downward, the corresponding part is ignored).

Such recommendations are obviously detailed and reasonable. Assumption: among all the

tags recommended in this way, there is a word (or a few) that appears very frequently in the tag, then check the highest-level tag that contains this word, and compare it to the next-level tag. Recommend to users together as implicit relevant knowledge.

In the recommendation process, if the user clarifies the category of knowledge (such as one or more restrictions in technology, equipment, model, etc., the relationship between the tag we recommend and its parent node must be the restricted category given).

After the tag recommendation, we will throw the next work to the crawler, because the tags we recommend are already in line with the knowledge domain and depth of knowledge that users want to understand to the maximum extent, so it is very easy to search using the recommended tags next. It is easy to get satisfactory results, here we can directly use the pagerank algorithm adopted by various search engines.

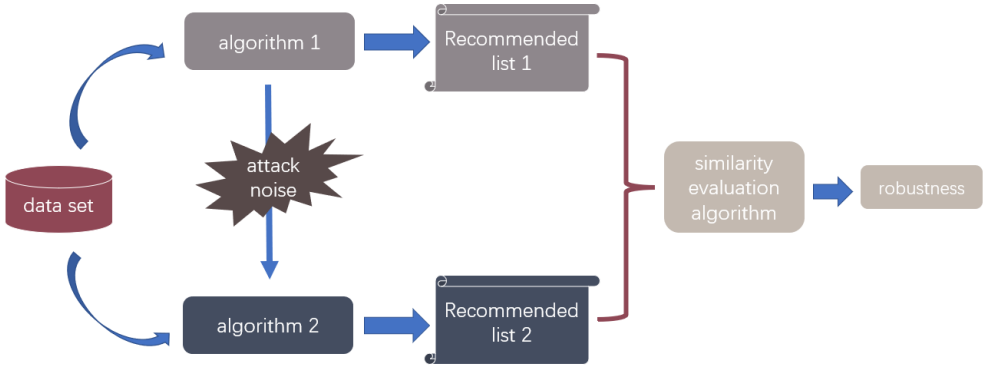

Figure 9: Robutness of the Knowledge Graph

Next, it is the evaluation model of the method performance of searching the knowledge item set under restricted conditions.

The model ***consists*** of the following parts:

1) Coverage(cov)

Definition: The **proportion** of "commodities/categories" that can be recommended by the recommendation system in the collection of "total products/categories". Assuming that the user set of the system is U, the recommendation system recommends a list of items $R(u)$ of length N to each user, and the total items are N. So:

$$\text{COV} = \frac{\sum R(u)}{N} \tag{7}$$

2) Gini coefficient(Gi)

The distribution reflected by the coverage rate is relatively limited. We can only know which categories are covered and which ones are not. Which category accounts for more and which category accounts for less? In order to describe the ability of the recommendation system

to discover the long tail in more detail, we need to **count the distribution of the number of occurrences of different categories in the recommendation list** and introduce the Gini coefficient to evaluate.

Definition: The Lorenz curve is drawn after the statistics are sorted according to the viewing times of the category from largest to smallest.

Taking the category distribution Gini coefficient as an example, to calculate the number of times all categories have been viewed, data statistics need to be performed in units of days.

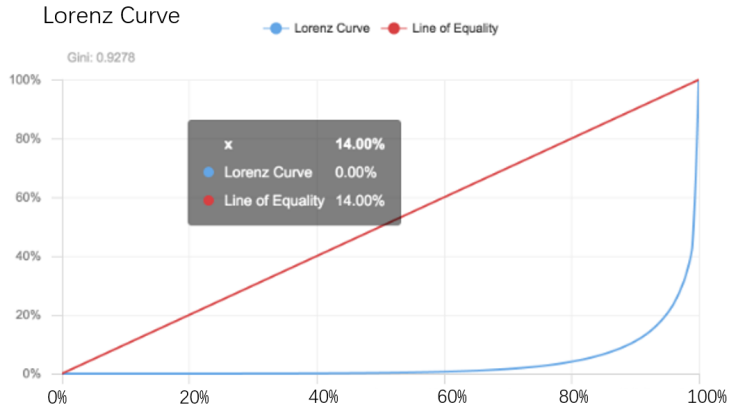

Figure 10: Gini coefficient based on Lorenz curve

Special note: the larger the Gini coefficient, the more uneven the distribution of all categories, the smaller the coefficient, the more even the category distribution. We know that every recommender system has its own categories, so the absolute average is not a good thing. The head category accounts for a little more but not too outrageous. For example: If we have 100 categories, It is relatively good that the top 5 accounts for $30-40\%$. Of course, it is not very meaningful to just look at this data in absolute terms. It is more about monitoring this indicator for a long time to see if there will be major changes.

3) Failure Rate(FR)

Definition: Indicates the proportion of data that the system does not recommend or is not clicked by the user after the recommendation in the complete set.

$$FR = \frac{S(0)}{S} \tag{8}$$

First, you need to define a time range to calculate the ones that are not recommended. Its meaning is the proportion of data that is not actually perceived by the user in the end. Unperceived includes content that has not been recommended and has not been clicked after being recommended.

4) Dispersion degree(D)

Definition: Describe the degree of dispersion of the result data in the recommended results.

$$\mathrm{D} = \frac{2}{k(k-1)} \sum_{i \neq j}^{k} 0.85^{\mathrm{Dis}((i,j)-1) * \mathrm{sim}(i,j)} \tag{9}$$

$$\begin{aligned} \text{if } i = j, \quad & \text{then } \mathrm{sim}(i,j) = 1 \\ \text{else}, \quad & \mathrm{sim}(i,j) = 0 \end{aligned} \tag{10}$$

It needs to be explained here. First, after calculating the scatter of two items (different positions), the overall scatter is obtained. The similarity function sim represents whether the pairs are the same. If they are the same, it is 1, and if they are not similar, it is 0. Regarding the influence of the distance between two contents on the degree of dispersion, it cannot be a linear relationship, because as the positions of the two products appear larger and larger, the users perception of the repeated products will gradually weaken (the closest position is There are two similar contents that may be a bit repetitive, but it is generally acceptable if there are two similar contents in a relatively distant location), generally there are about 4 contents on the dual-stream screen, $0.85^{(5-1)}$ is about It is about 0.5, so if it is within 5, the dispersion will be very low, but if it is> 5, the dispersion will not attenuate much. The closer two similar items are, the greater the weight.

4.5 Sub-model 4: Feedback Learning Model

We build the adaptive learning and optimization of the knowledge graph system from the following parts:

1) Use neural network to change some parameter values ($rank$, D, Dis, etc.) through user feedback (implicit feedback mainly).

2) Extract keywords (tags) of the newly added knowledge system and establish new relationships (incorporate into the original structure).

3) Re-associate, compare, and trade off the new relationship generated by the original node.

Next we will elaborate on the above three parts in detail.

- We use convolutional neural networks to train parameter values through user feedback values. The specific model is a dynamic programming model based on a convolutional neural network. First of all, our feedback data are the evaluation parameters in the model overview and model 3, including:Coverage rate cov, Gini coefficient Gi, Failure rate FR, Dispersion D. Second, our training parameters are mainly $rank[]$ array, generalized distance D and narrow distance Dis. The algorithm we use supports stochastic gradient descent and batch gradient descent, the activation function supports *sigmoid*, the output layer is *softmax*, the pooling core supports *average*, and the weight initialization uses the *Xavier* method. (See appendix for python code)

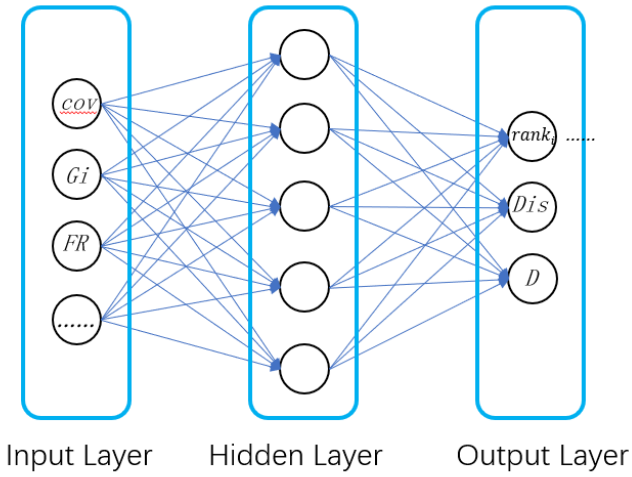

Figure 11: Convolutional Neural Network

- For the newly added knowledge system, we use data crawling to obtain, and then perform data pre-filtering, and then use the CRF algorithm based on the NLP field to extract keywords from the text information, and the unsupervised syntax analysis method is used to complete the device relationship identification.

  First, we define the degree of relevance:

  Definition: (Relevance R) In the existing nodes, the correlation between two nodes.

$$R(x,y) = \frac{D(x,y)}{\mathrm{Dis}(x,y)} \tag{11}$$

  For a new label, first calculate the correlation between the new node and the existing node, and then normalize the distance to get the distance. The basic process is to traverse the original node after pruning, using unsupervised grammar each time The analysis method is processed again, and the result is normalized with the defined correlation. Then you can use the obtained correlation value to calculate the distance The relationship is as follows:

$$D(x,y) = \mathrm{Dis}(x,y) \cdot R(x,y) = |\mathrm{rank}_x - \mathrm{rank}_y| \cdot R(x,y) \tag{12}$$

  Since $rank_y$ is known to us, we might as well set $rank_x$ definition as the geometric mean of $rank$, because although there may be deviations, this is already the solution with the smallest variance, and the convolutional neural network in 1 will train and correct the rank value. The process of relationship recognition is to extensively associate new tags with existing tags, but the knowledge graph formed by this is only a graph, and is not reduced to a tree. Therefore, we will use the dotted line to connect the new label with the existing label and calculate a correlation value through the above algorithm, and then determine several solid lines among the many dotted lines so that the solid lines form a minimum spanning tree. Considering that our graph is edge-dense, we use the ***Prim*** algorithm to create a minimum spanning tree

for the reciprocal of the correlation degree, that is, the shortest path traversed by the ***Prim*** algorithm is changed from a dotted line to a solid line.

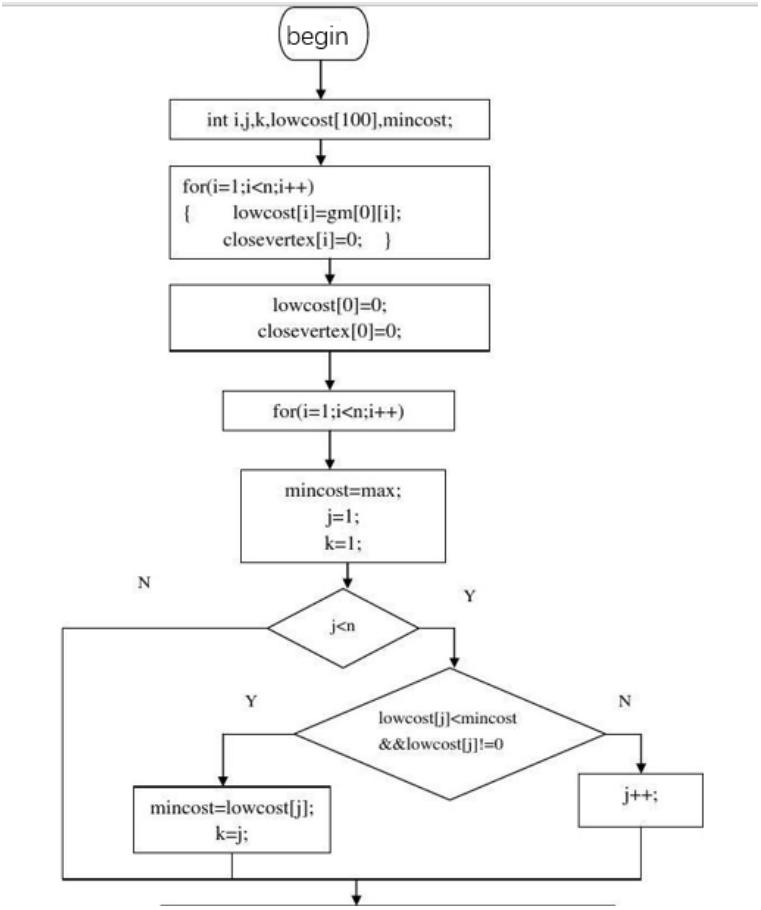

Figure 12: Block diagram of Prim algorithm

- For the situation that the existing node has a new association or a generalized distance change, the two endpoints of the changed edge are taken as the center, and the local ***Prim*** algorithm is performed separately. If the ***Prim*** algorithm passes the dotted line, the dotted line is changed. It is a solid line, and then change the original solid line to a dashed line.

5 Model Solution

5.1 Model 1 Solution

First we need to generate knowledge tree tags.The data in Data.csv is processed as follows:

- First convert the Excel format file into a comma-separated file format ($.CSV$ file)
- Use the csv library function $csv.reader()$ in python to read the converted $.csv$ file

- Call the $Node()$ function to store the value of each column in the same node $start\_node$, $relation$,end_node
- **Use $jdk11$ version of JAVA to be compatible with $neo4j-4.3.7$ version, use py2neo's $create()$ library function to generate nodes, use neo4j community version to draw nodes and realize automatic generation of hierarchical relationships and visualization operations.**

For the five tags given, we use the Excel keyword search method to extract all the items in the Data data that contain one of the keywords, and use the same method as the above to generate a map to generate a map, that is, a sub-tag The tree structure is complete, and the remaining four tags are completed by analogy.

Two of the label generation results are shown in the figure:

Figure 13: Local search

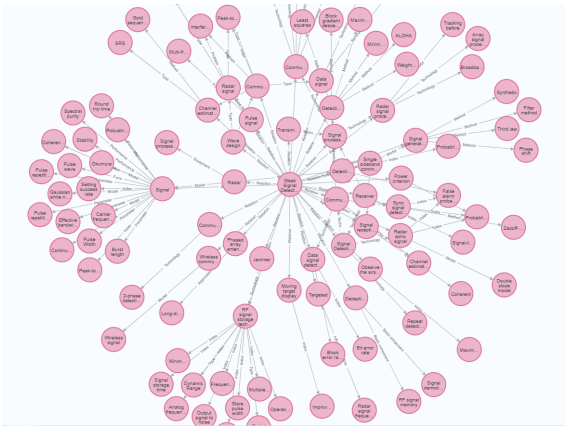

Figure 14: Global search

Figure 15: Weak signal detection

Figure 16: Local search

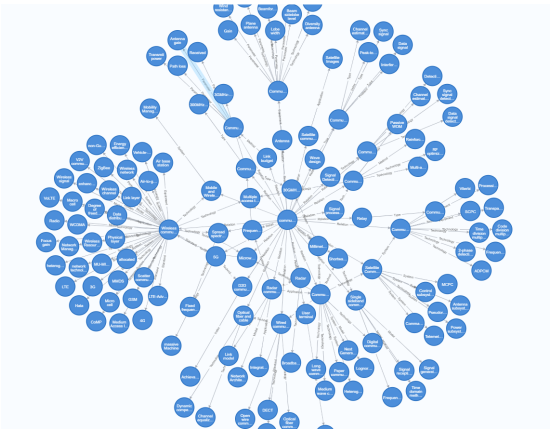

Figure 17: Global search

Figure 18: Communication under complex electromagnetic environment

5.2 Model 2 Solution

Regarding the authoritative academic paper search website as the database and referring to the evaluation indicators in the information retrieval model, we use the same two indicators R and P to evaluate the accuracy and efficiency of the system respectively, and then combine the two indicators in a weighted average To get the index F together, we give the formula of F. The closer the score of F is to 1, the better the system.

5.3 Model 3 Solution

In the search process, we use the shortest path optimization algorithm that is Dijkstra's algorithm combined with heap optimization to calculate the similarity between nodes to achieve the recommendation of similar node information.

Dijkstra algorithm can be simply understood as *breadth first search (BFS) plus greedy algorithm*, because it starts from the source point and searches for the shortest point in the surroundings, and then continues searching from the adjacent shortest point to the surroundings, and finally finds out from The shortest path from the source point to the end point.

Our heap optimization greatly improves the efficiency of the algorithm, and the time complexity is $O(NlogN)$ (see the appendix for the source code)

The performance comparison is shown in the figure, the above is the ordinary Dijkstra algorithm, and the following is our heap optimization:

Figure 19: Heap optimizationm

5.4 Model 4 Solution

We use a **convolutional neural network that supports stochastic gradient descent and batch gradient descent**. In terms of parameter settings, the activation function we use is a typical function model *sigmoid*, the output layer is *softmax*, the pooling core supports *average*, and the weight initialization uses the *Xavier method*. (See appendix for specific code)

The training result of the Rank parameter is shown in the figure:

The vertical axis represents Loss, and the horizontal axis represents Batches. The effect is relatively stable.

When generating a new relationship, we use the *Prim algorithm* to change the shortest path it travels from a dotted line to a solid line.

The results are good. The test generation diagram is as follows:

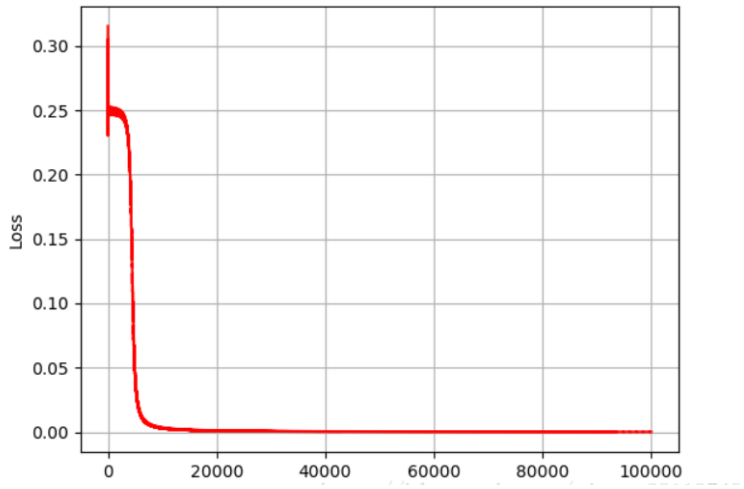

Figure 20: Training result of Rank parameter

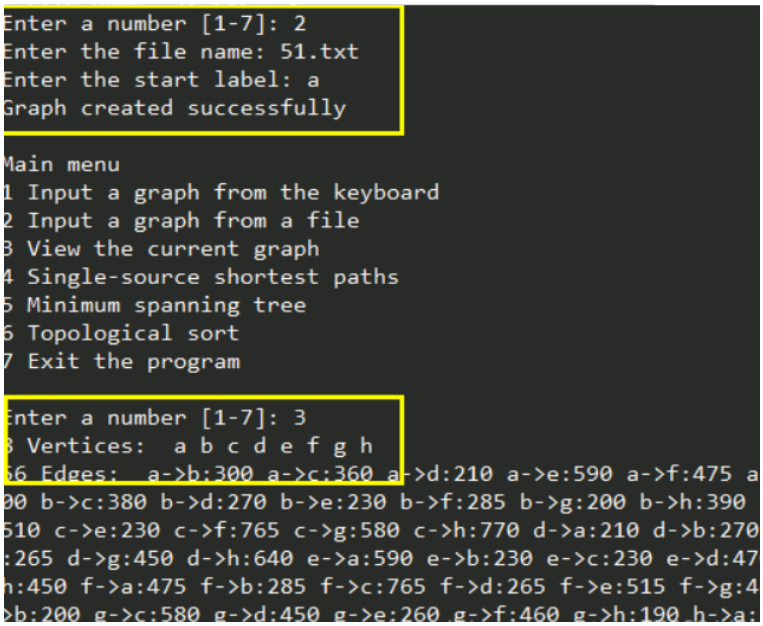

Figure 21: Use Prim algorithm to test the generated graph

6 Evaluation of Modeling Results

- The knowledge tree model that we draw by code has the advantage of using images to directly show the distance and logical relationship between knowledge nodes. Various functions can be realized with the help of Neo4j. The disadvantage is that it is more restricted by the platform and certain operations, which means not free enough.

- In the search and recommendation model, the advantage of this model is that it can output the data set in a logical order. The disadvantage is that the user imposes less restrictions on the scope of the item, and there is no option about the order relationship of the output content,

such as character length sorting, initial letter sorting, etc. The commonly used methods are not considered within the scope of this model.

- In the feedback learning model, the advantage is that it not only considers the influence of the user's subsequent choices, but also absorbs new knowledge, thus conforming to the development of the times and science. The disadvantage is that it does not consider the operating efficiency of the system, and may run slowly when processing large or complex data.

7 Appendix

```python
import csv
import py2neo
from py2neo import Graph,Node,Relationship,NodeMatcher
g=Graph('neo4j://localhost:7687',user='neo4j',password='1234')
with open('C:/Users/rainsound/Desktop/Data.csv','r',encoding='utf-8') as f:
    reader=csv.reader(f)
    for item in reader:
        if reader.line_num==1:
            continue
        print("当前行数：",reader.line_num,"当前内容：",item)
        start_node=Node("Field",name=item[0])
        end_node=Node("Field",name=item[2])
        relation=Relationship(start_node,item[1],end_node)
        g.merge(start_node,"Field","name")
        g.merge(end_node,"Field","name")
        g.merge(relation,"Field","name")
```

```python
def span_tree(g, start_label):
    """最小生成树"""
    min_weights = 99999
    results = []

    def recurse(w, stack):
        if w.is_marked() == False:
            # 顶点未标记则入栈
            stack.push(w)
            w.set_mark()
            # 未遍历完的情况
            if stack.get_size() != g.size_vertices():
                for x in w.neighboring_vertices():
                    # 继续从顶点的连通节点往下继续遍历
                    recurse(x, stack)
                    # 如果从w的连通节点再往下没有连通节点，
                    if stack.peek() != w and \
                            stack.get_size() < g.size_vertices():
                        temp = stack.pop()
                        temp.clear_mark()
            else:
                # 已经遍历完所有顶点，并且满足了条件，
                # 但最后一个节点还有连通的顶点时，不再遍历
                if w.get_edge_count() > 0:
                    return None
                pass
        else:
            return None
```

```
    def generate_results(sun_stack):
        temp = ""
        for v_temp in sun_stack:
            temp += v_temp.get_label() + "->"
        temp = temp[:len(temp) - 2]
        temp += ": " + str(sun_stack.get_weights())
        temp += "\n"
        results.append(temp)
    for v in g.vertices():
        stack = LinkedStack()
        stack.push(v)
        sun_stack = copy.deepcopy(stack)
        for w in v.neighboring_vertices():
            g.clear_vertex_marks()
            v.set_mark()
            recurse(w, sun_stack)
            if sun_stack.get_size() == g.size_vertices():
                if sun_stack.get_weights() < min_weights:
                    results.clear()
                    generate_results(sun_stack)
                    min_weights = sun_stack.get_weights()
                elif sun_stack.get_weights() == min_weights:
                    generate_results(sun_stack)
            else:
                pass
            sun_stack = copy.deepcopy(stack)

    if results == None:
        return "No best path"
    string = "Here are " + str(len(results)) + " path:\n"
```

```
    results.insert(0, string)
    return results
```

```
import numpy as np
import matplotlib.pyplot as plt
from convolutional_neural_network import cnn


def config_net():
    """
    配置网络。
    """

    # 输入层
    size_input = np.array([8, 8])
    args_input = ("input", (size_input,))

    # C1卷积层
    connecting_matrix_C1 = np.ones([1, 2])
    size_conv_kernel_C1 = np.array([5, 5])
    stride_conv_kernel_C1 = 1
    padding_conv_C1 = 0
    type_activation_C1 = "sigmoid"
    args_C1 = ("convoluting", (connecting_matrix_C1, size_conv_kernel_C1,
                               stride_conv_kernel_C1, padding_conv_C1,
                               type_activation_C1))
```

```
    # S2池化层
    type_pooling_S2 = "average"
    size_pool_kernel_S2 = np.array([2, 2])
    stride_pool_kernel_S2 = 2
    padding_pool_S2 = 0
    type_activation_S2 = "sigmoid"
    args_S2 = ("pooling", (type_pooling_S2, size_pool_kernel_S2,
                           stride_pool_kernel_S2, padding_pool_S2,
                           type_activation_S2))

    # C3卷积层
    connecting_matrix_C3 = np.ones([2, 2])
    size_conv_kernel_C3 = np.array([2, 2])
    stride_conv_kernel_C3 = 1
    padding_conv_C3 = 0
    type_activation_C3 = "sigmoid"
    args_C3 = ("convoluting", (connecting_matrix_C3, size_conv_kernel_C3,
                               stride_conv_kernel_C3, padding_conv_C3,
                               type_activation_C3))
```

```
    # 输出层
    n_nodes_output = 2
    type_output = "softmax"
    args_output = ("output", (n_nodes_output, type_output))

    args = (args_input,
            args_C1,
            args_S2,
            args_C3,
            args_output)
    cnn_net = cnn()
    cnn_net.config(args)

    return cnn_net
```

```
n_train = 10000
X_train = 0.2 * np.random.randn(8, 8, n_train)
Y_train = np.random.randint(2, size=n_train)
for i in range(Y_train.shape[0]):
    if Y_train[i] == 0:
        X_train[1, :, i] += np.ones(8)
    elif Y_train[i] == 1:
        X_train[:, 1, i] += np.ones(8)

size_batch = 50
n_epochs = 500
cnn_net = config_net()
cnn_net.fit(X_train, Y_train, size_batch=size_batch, n_epochs=n_epochs)

n_test = 1000
X_test = 0.2 * np.random.randn(8, 8, n_test)
Y_test = np.random.randint(2, size=n_test)
for i in range(Y_test.shape[0]):
    if Y_test[i] == 0:
        X_test[1, :, i] += np.ones(8)
    elif Y_test[i] == 1:
        X_test[:, 1, i] += np.ones(8)

correct_rate = cnn_net.test(X_test, Y_test)
```

```
plt.figure()
for i in range(cnn_net.layers[1].n_nodes):
    plt.subplot(1, 2, i + 1)
    plt.imshow(cnn_net.layers[1].nodes[i].conv_kernels[0], cmap="gray")
plt.show()
```

```
def dijstra(adj, src, dst, n):
    dist = [Inf] * n
    dist[src] = 0
    book = [0] * n# 记录已经确定的顶点
    # 每次找到起点到该点的最短途径
    u = src
    for _ in range(n-1):    # 找n-1次
        book[u] = 1# 已经确定
        # 更新距离并记录最小距离的结点
        next_u, minVal = None, float('inf')
        for v in range(n):    # w
            w = adj[u][v]
            if w == Inf:    # 结点u和v之间没有边
                continue
            if not book[v] and dist[u] + w < dist[v]:# 判断结点是否已经确定了,
                dist[v] = dist[u] + w
                if dist[v] < minVal:
                    next_u, minVal = v, dist[v]
        # 开始下一轮遍历
        u = next_u
    print(dist)
    return dist[dst]
```

References

[1] Hu Sen,Zou Lei,Yu Jeffrey Xu,Wang Haixun,Zhao Dongyan. Answering Natural Language Questions by Subgraph Matching over Knowledge Graphs[J]. IEEE Transactions on Knowledge and Data Engineering,2018,30(5)

[2] P. Zhao and J. Han, On graph query optimization in large networks, Proceedings of the VLDB Endowment, vol. 3, no. 1-2, pp. 340351, 2010.

[3] X. Wu and S. Deng, Research on optimizing strategy of database-oriented gis graph database query, in 2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS). IEEE, 2018, pp. 305 309

f11.m 2-Dijkstra.m

```
function [mydistance,mypath]=Dijkstra(a,sb,db)
%a为邻接矩阵 a(i,j)为距离 sb db为起点和终点 输出最短路距离和最短路路径
%这里可以根据知识图谱的大小，输入邻接矩阵的参数
%先初始化a=zeros(n),再输入权a(i,j)=λi*f(time-j)
n=size(a,1); visited(1:n)=0;
distance(1:n)=inf; distance(sb)=0;
visited(sb)=1; u=sb; parent(1:n)=0;
for i=1:n-1
    id=find(visited==0);
    for v=id
        if a(u,v)+distance(u)<distance(v)
            distance(v)=distance(u)+a(u,v);
            parent(v)=u;
        end
    end
    temp=distance;
    temp(visited==1)=inf;
    [t,u]=min(temp);
    visited(u)=1;
end
mypath=[];
if(parentdb)~=0
    t=db; mypath=[db];
    while t~=sb
        P=parent(t); mypath=[P mypath]; t=p;
    end
end
mydistance=distance(db);
```

Figure 22: BFS