2022 年美国数学建模竞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

我们报名**队伍**号是：H15

参赛队员姓名学号（**中文填写打印并签名**）：

1.孙启皓 2020190904012

2.徐源成 2019100403021

3.陈禹桥 2020190901025

指导教师或指导教师组负责人（有的话填写）：王志勇

是否愿意参加 2022 年美国赛（是，否）：是

日期：2021 年 11 月 28 日

Adaptive Knowledge Graph System Covering Four Aspects and Its Visualisation

Summary

Our Knowledge Graph (KG) system is essentially a self-adaptive semantic retrieval system including communication, reconnaissance, detection and interference. Therefore, it can be used to query no matter what items, and give the outcomes as precise as possible even if the input is relativelt fuzzy.

For Question 1, we define labels as subclasses and items as instances. We extracted labels and items from attachment, encyclopedia websites, and academic libraries and structure them in the same form. After filter and combination, the subclass-of relations (relations between two labels) and the instance-of relations (relations between items and labels) are calcuated by using NLIP. In Section 2.3, two types of labels (electromagnetic interference & signal identification under strong interference) are completed and taken as examples. Finally, a System Evaluation Model with three quality dimensions and twelve indexes is constructed to evaluate the performance of our system. Our system perform very well in all dimensions comparing to encyclopedia websites and academic libraries.

For Question 2, the solution we done is essentially an algorithm of semantic retrieval. In our Knowledge Items Connection Model, condidate items are chosen to connect to new items with semantic information under the rules we gave in Section 3.1. We adopt BERT and CRF to extract items from input document and joint code the items with existing items in our system. However, BERT and CRF do not perform well in identifying mathematical items. Therefore, we additionally adopt Mathpix to transform mathematical items into strings in LaTeXgrammer, which can be understood by computer.

For Question 3, we built the Two-staged Top-k Query Model to keep precisely query under constricted conditions. In Sectio 4.3, four creterions of model evaluation are given and our model perform the best among three existing algorithms.

For Question 4, based on machine learning algorithm, we constructed the Self-adaption Algorithm. This algorithm uses multiple potential vectors and can help the system predict one of the element in a triple. After verifying, it successfully predicted the missing data in test sets.

For Question 5, we designed a visualized software with layer architecture. From Section 6.2 to 6.3, we have given examples of different situations of using the software. In Section 6.3, the simplified presentation of our software ad knowledge graph system is shown.

In summary, we fully completed the tasks in the problem and the system performs very well in various types of evaluation.

Key words: Knowledge graph, Machine learning, Natural Language Processing

Contents

1 Introduction

1.1 Background

Knowledge graph (KG) is first put foward by Google in 2012 and rapidly attract popularities from then on[1, 2]. Considerable research into knowledge graphs has been carried out in recent years, especially in the Semantic Web community[1]. Therefore, a variety of partially contradicting definitions and descriptions has emerged[1]. Generally, knowledge graph is a knowledge base that uses a graph-structured data model or topology to integrate data[3].

With the exponential increase of data in the Web community, traditional searching methods are not efficient and comprehensive enough to meet the need of users[2]. This problem has been improved by knowledge graphs. It can efficiently explore and analyze the connection between pieces of knowledge, and construct complex knowledge webs[2, 4]. Through visualization, the relationships between knowledge can be clearly presented, which is beneficial to knowledge fusion and interdisciplinary researches[4].

Communication, reconnaissance, detection and interference are four overlapping fields with deep connections. Research into these fields has been very deep and mature, but it is still relatively difficult to do knowledge fusion since the relationships between them are still unclear[4]. By constructing knowledge graph system, knowledge in these fields could be reorganized and both their similarities and differences could be explored[2, 3, 4]. This would greatly contribute to researches including but not limited to interdisciplinary technologies and their application[1].

1.2 Problem restatement

Authors believe the problem is actually to construct a semantic retrieval system which has ability to self-adaption. Therefore, we restate the questions as followings based on researches in order to simplify the problem.

Question 1: Authors believe that to constructing a knowledge graph system is to design the classification scheme with labels, items, and relations. Labels are called as subclass while items are called as instance in most research. The relations can be seperated into two types: relations between labels(subclass-of relations) and between items and labels (instance-of relations). To evaluate a knowledge graph system is actually test its ability of semantic searching. Thus, to compare it with traditional searching methods is what we should do.

Question 2: To connect items by using obtained labels can be seperated into two aspects. First, connect and combine the items with similar labels and reoccuring semantic information. However, this aspect is unnecessary if the classification scheme in Question 1 is efficient enough. A second aspect is to connect the items from outside the system and the items inside the system by using obtained labels. Therefore, the rules to connect obtained items to the input items, namely key word retrieval rules, should be given.

Question 3: The knowledge graph can be regarded as an undirected graph, and the user's query can also be seen as an undirected graph, denoted as a query graph. Constraints are the restricted conditions that are given in advance by the user when querying. We can use these constraints to search in the knowledge graph and get a subset of the knowledge graph as a response. In addition, we need to give certain evaluation criterion when comparing with traditional algorithms.

Question 4: This question is actually to design a mechine learning algorithm. For a self-adaption knowledge system, the ability it should have is to dig out the potential knowledge in the knowledge graph system. To improve the content, the system should have the ability to predict potential pieces of knowledge by using existing pieces of knowledge. To improve the structure, the system should have the ability to predict the relations between two pieces of knowledge.

Question 5: Give clear structural framework and simplified presentation system.

1.3 Planned approach

To solve the problem, authors plan to use approaches presented below.

Question 1: As mentioned in Section 1.2, to condtructing a knowledge graoh system is to design a classification scheme with labels, items, and relations. Therefore, we plan to first extract labels and items from the online databases and transfrom them into a same structure. Then, their relations can be explored by using algorithm like Non-linear Interger Programming (NLIP). The evaluation model can have several dimensions to evaluate the system's performance comparing to traditional databases.

Question 2: To connect items, we plan to design a model which can automatically find the items that are possible to connect to items in given document by obeying several rules. Algorithms like BERT, CRF, and Bi-LSTM are planned to use in this model to achieve semantic identification. For mathematical items and figures, BERT can not extract them very well, therefore we plan to use Mathpix to help the system identify the mathematica items and figures.

Question 3: As the knowledge graph is complex and has noisy data, we need to choose an algorithm that can figure out the topological features clearly. Thus, we plan to optimize the traditional Top-k Algorithm and then get a two-staged version of it. Thereafter, we will choose several traditional algorithms to compare them with our model according to the given criterion.

Question 4: As mentioned in Section 1.2, a machine learning algorithm need to be designed. Therefore, we plan to design a machine learning algorithm on the basis of mature algorithms in other fields like Zero-shot Learning Algorithm.

Question 5: We plan to design a visualized software to implement functions.

This paper is organized as Figure 1:

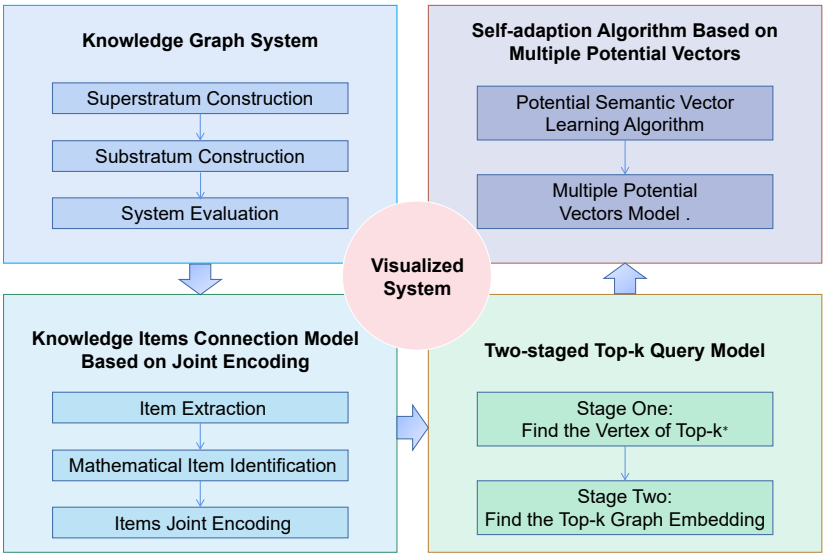

Figure 1: Order of this paper

2 Knowledge Graph System

2.1 Superstratum Construction

The superstratum construction of a knowledge graph system are actually to design labels and items, filter and combine labels, and to construct subclass-of relations.

2.1.1 Item Extraction and Structuring

There are three sources to be used to extract items. First, we use the attached file (Data.xlsx) to extract items, which are however not comprehensive enough and lack of semantic information. Second, we extract items from encyclopedia websites. Baidu Baike and Wikipedia are chosen to not only explore new items, but also complete the semantic information of items that have been extracted. Since the amount of work is impossible to be finished by human, crawler is used in this process. Due to the websites are not for professional fields, therefore they cannot meet our need perfectly. Finally, we choose professional e-documents from University of Glasgow Library and Google Scholar. Since these libraries block crawler access, authors manually filter documents to provide them for crawlers. After that, we obtain the names and semantic information of items.

To structure the items, we regard an item as an object with six element, which contains its title, set of its inner links, set of its catalogue, set of its property, word set it associate with, and set of it tags. This can be expressed as $Item(a) = \{T(a), L(a), C(a), P(a), R(a), H(a)\}$.

2.1.2 Label Extraction and Structuring

We structured items in the previous section, but items are merely the leaf nodes in the knowledge graph system while the labels are the backbones nodes. First, we choose Electronics and Information System (F01) of the categories from Natural Science Foundation of China (NSFC) to be the first three layers of our label structure. Then, according to common sense, we extract labels from the tags of encyclopedia websites and academic libraries. These tags cover a wild range of labels, but many of them are not precise labels and contain a part of items. Thus, we need to filter and combine the labels we extracted, which will be in the next section.

Apart from extraction, we also need to structure the labels. Similar with item structuring, we can regard each label as an object with six element, which contains its title, the items it associate with, set of its inner links, set of its catalogue, set of its property, and word set it associate with. This can be expressed as $Label(h) = \{T(h), L(h), C(h), P(h), R(h), A(h)\}$ and the calculation equations are as following:

$$A(h) = \{\cup a, h \in H(a)\}$$
$$L(h) = \{\cup L(a), a \in A(h)\}$$
$$C(h) = \{\cup C(a), a \in A(h)\}$$
$$P(h) = \{\cup P(a), a \in A(h)\}$$
$$R(h) = \{\cup R(a), a \in A(h)\}$$

2.1.3 Subclass-of Relations Construction

Since the amount of labels is relatively small comparing with items, we use a bottom-up method to construct subclass-of relations. This means that we should find possible superclass of each label and adjust the validity of the relation. The method we use to find possible superclass is to compare the sets of items which the labels associate with, namely their $A(h)$. If two labels have similar $A(h)$, they have relatively high possibility to construct subclass-od relation. We can judge the direction of subclass-of relation through comparing the size of the labels' $A(h)$, and the label with bigger $A(h)$ is superclass.

2.1.4 Label Filter and Combination

The label filter and combination is done at two stages. Before we construct subclass-of relations, we should eliminate the labels which associate with few items, which is because that a label associated with few item have a high possibility to be an item. After we construct subclass-of relations, we should combine the redundant labels and eliminate the isolated labels.

2.2 Substratum Construction

The substratum construction of a knowledge system is actually to connect items to labels through instance-of relations.

2.2.1 Instance-of Relations Consruction

Since the amount of items is too big to go through all labels for each item, we choose several labels to be chosen for an item and judge if the instance-of relation is valid. After research, we choose to use Non-linear Interger Programming (NLIP) [5].

To express the relevancy between item a and label h, the relevancy function is:

$$coh(a,h) = \frac{1}{1+\mu^{\sum w_i \cdot l_i} \cdot \sum w_j \cdot d_j}$$

$coh(a,h)$ represents the relevancy, l_i and d_j represent the two morphology characteristics and the four structural characteristics among the six charateristics mentioned in Section 3.1.1, w_i and w_j represent the weight of characteristics.

Moreover, the instance-of relation should not collide with the subclass-of relations we obtained. Therefore, the equation is as following:

$$\max \sum_{h_j \in H_a} \left[|H_a| \cdot y_j \cdot coh\left(a, h_j\right) + \lambda \cdot \sum_{h_k \in H_a} y_k \cdot \left(1 - y_j\right) \cdot \left(1 - coh\left(h_k, h_j\right)\right) \right]$$

$$\text{s.t. } \forall h_j \in H_a, \qquad y_j \in \{0,1\},$$

$$\forall \left\langle h_j, h_k \right\rangle \in \{h_j \text{ is subclass-of } c_j\}, \qquad y_k - y_j \geq 0$$

When two labels h_k and h_j are mutex, the weaker their relevancy is, the stronger punishment is; the stronger their relevancy is, the weaker punishment is.

We use the IBM CPLEX Optimizer to solve the equations.

2.3 Methods Application and Examples

After applying our methods and constructing the knowledge graph, we choose electromagnetic interference and signal identification under strong interference two kinds of labels to be examples. We present their branches as Figure 2.

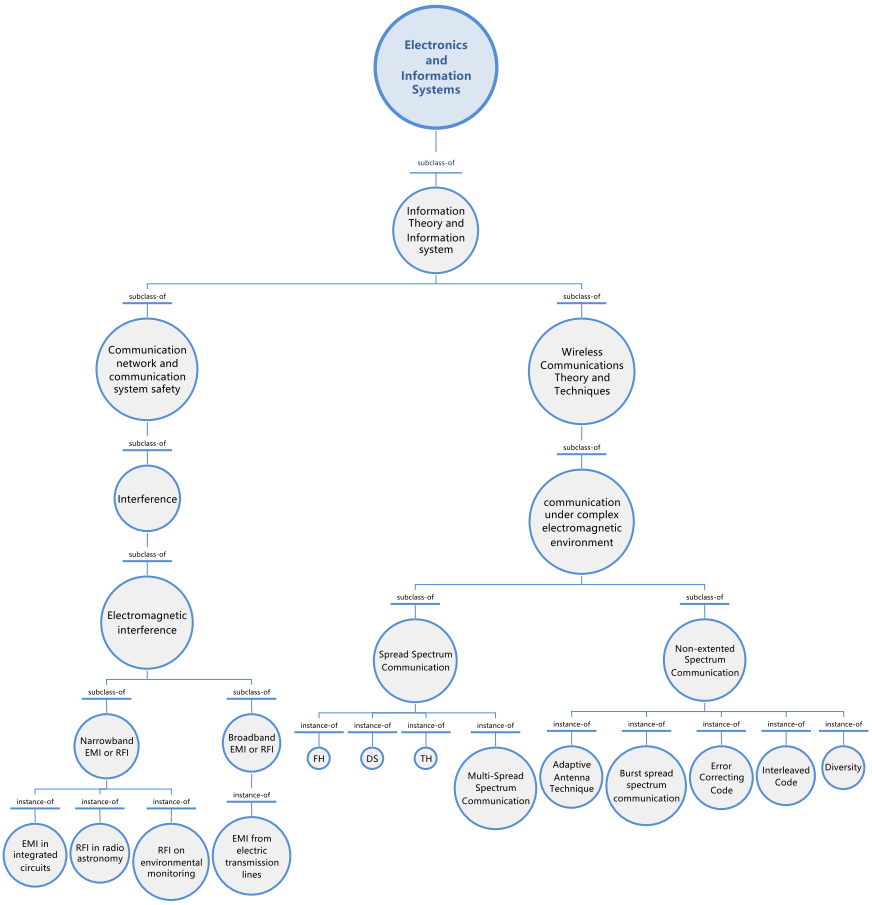

Figure 2: Branches of electromagnetic interference and signal identification under strong interference

2.4 System Evaluation Model

Bases on the quality dimensions in [6], we focus on the process of generating the system to adjust quality dimensions of the system and give three types of quality dimensions: credibility, resource description, and redundancy. The detailed dimensions in each type are shown as Figure 3.

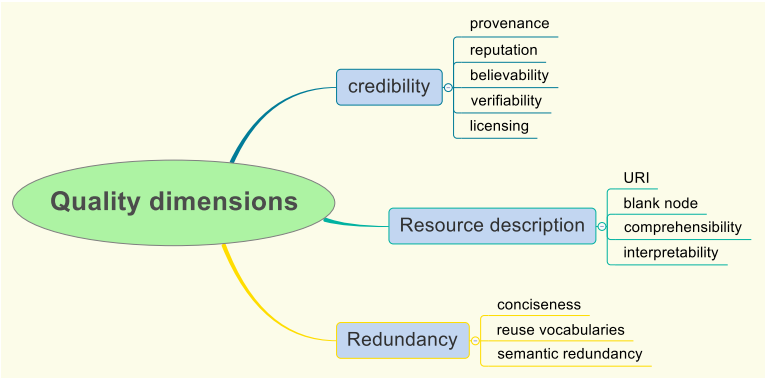

Figure 3: Detailed quality dimensions of system evaluation

2.4.1 Credibility Evaluation

For credibility evaluation, we adapt Dataset Ranking Algorithm given by Delbru[7].

First, calculate the weight of links in datab sets, and the equation is:

$$w_{\sigma,i,j} = \frac{\left|L_{\sigma,i,j}\right|}{\sum_{L_{\tau,i,k}} \left|L_{\tau,i,k}\right|} \times \log \frac{N}{1 + \text{freq}\,(\sigma)}$$

$|L_{\sigma,i,j}|$ represents the relations between links in data set i and j, $\sum_{L_{\tau,i,k}} \left|L_{\tau,i,k}\right|$ represents the amount of link relations from i to other data sets, N represents the amount of data sets, freq(σ) represents how many times σ occurs in all data sets. On the basis of $w_{\sigma,i,j}$, we can get the ranking equation of data sets among all data sets:

$$r^k\left(D_j\right) = \alpha \sum_{L_{\sigma,i,j}} r^{k-1}\left(D_i\right) w_{\sigma,i,j} + (1-\alpha)\frac{\left|E_{D_j}\right|}{\sum |E_D|}$$

Second, after calculating the rank of each data set, the rank of an item e can be calculated:

$$r(j) = \sum_{j \in D_k} r(D_k)$$

Third, considering both the ranks of data sets and the ranks of items, we can use the average rank of all items in a data set to be the rank of the data set:

$$Rank = \frac{\sum_{j \in \{1,2,\ldots,N\}} r^k\left(D_j\right) \times r(j)}{N}$$

2.4.2 Redundancy Evaluation

We can use $|F|$ to represent the size of the system, r_b to represent the average bytes of every node in the system, and N_c to represent the total times all nodes occur. Then we can get the equation:

$$|F| = r_b \times N_c$$

If there exists another form of the system which can make r_b or N_c smaller, then the system has structual redundancy and the equation is:

$$R_{structure} = \frac{|\bar{F}|}{|F|}$$

$|\bar{F}|$ is the size of the system after changing form.

If there are two nodes in the system, N_1 and N_2, and N_1 has redundant triples, then the system has

semantic redundancy and the equation id:

$$R_{semantic} = \frac{|D_2| + |R|}{|D_1|}$$

$|D_1|$, $|D_2|$ means the amount of triples in their nodes and $|R|$ means the amount of redundant triples.

To combine the redundency above, the totak redundancy is:

$$R = \sqrt{\frac{R^2_{structure} + R^2_{semantic}}{2}}$$

2.4.3 Resource Description Evaluation

Since there is no standard criterion for this dimension, this dimension should be evaluated manually.

2.4.4 Evaluation Result

After comparing our system to Baidu Baike as resprent of encyclopedia websites and University of Glasgow Library as resprent of academic libraries, the result is as Table 1.

Table 1: Evaluation result of the knowledge graph system

	KG system	Baidu Baike	UoG Library
Credibility	Medium	Poor	Good
Redundancy	Good	Poor	Medium
Resource description	Good	Poor	Good

Table 1 shows that the KG system has good redundancy and resource decription, but is slightly limited in credibility. Authors believe that the reason is that we adopted encyclopedia websites to construct labels and items. In general, the knowledge graph system performs very well comparing to encyclopedia websites and academic libraries.

3 Knowledge Items Connection Model Based on Joint Encoding

According to Section 1.2, we are supposed to give rules to connect input items to the items in the system. Assumes that a piece of document D is given, which contains a set of words $D = \{w_1, w_2, \ldots, w_L\}$, and w_i is the i-th word in the document. To connect items mentioned in D to the knowledge graph system, we should first extract a set of items in D: $M = \{m_1, m_2, \ldots, m_N\}$, and m_j is the j-th item mentioned in D. Then, we should search the set of all candidate items in the system for each m_j: $C_{m_j} = \{e_1, e_2, \ldots, e_M\}$, and e_k is the k-th candidate item. To select candidate items, authors give four rules as following:

1. The title of candidate item contains all or part of the title of mentioned item;
2. The title of candidate item is exacly the capital letters of all words in the title of mentioned item;
3. The title of candidate item shares several words in same with the title of the mentioned item;
4. The title of candidate item has high similarity with the title of mentioned vector.

Any item in the system should be selected as candidate vector if it satisfies any one of the rules. Then, we should calculate the similarity of each e_k and m_j in the set C_{m_j}, and judge whether the candidate item e_k is the aim item mapped by mentioned item m_j. Figure 4 shows the whole process of item connection and two different models.

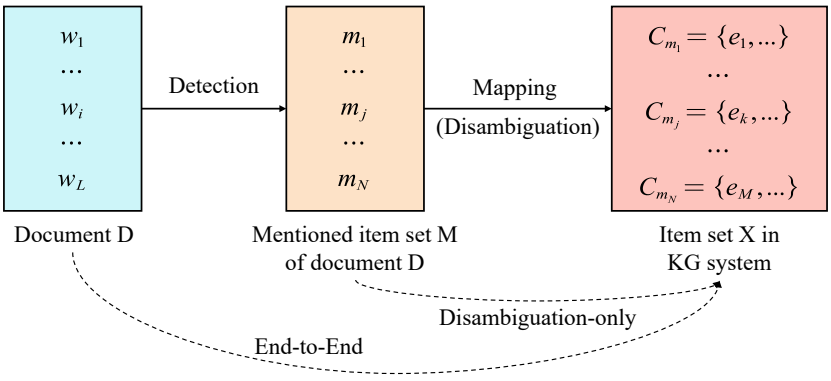

Figure 4: Process of item connection

3.1 Model Struture

The model structure has been shown as Figure 5. The input of the model is the original given document, while the output are many triples, which consist of mentioned item, location, and labels. This end-to-end method reduces the propagation of error, and enhances the connection between item extraction and item disambiguation. Meanwhile, the model joint codes on both mentioned items and candidate items, which can start to learn the connectopn between items from encoding.

Initially, the model codes the original document through Bidirectional Encoder Representation from Transformers (BERT), then decodes to obtain mentioned items through Conditional Random Field (CRF). After that, model uses sharing BERT to joint code mentioned items and candidate items. Next, model inputs the output of BERT into bi-directional Long Short-Term Memory (Bi-LSTM) to extract labels, and seperate the output into mentioned vectors and average vectors. Following that, the mentioned vectors and the average vectors are linked and input into fully connected layer to judge whether the connection between metioned items and candidate items is valid. Finally, the model outputs the triples of mentioned item, which includes mentioned item, position, labels.

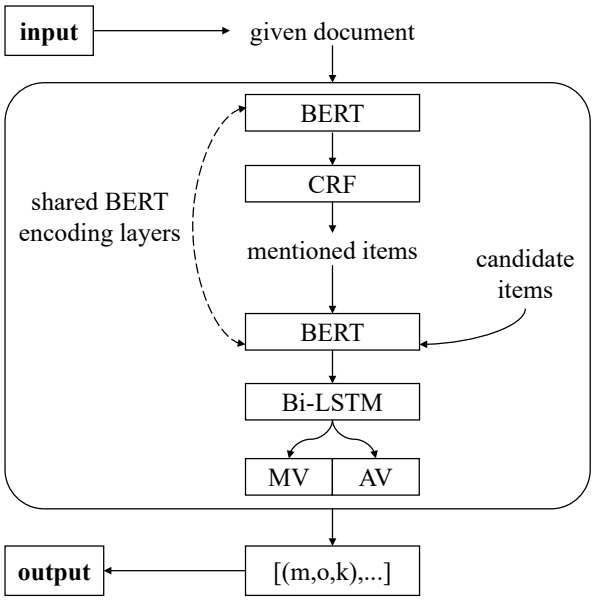

Figure 5: Model structure of Knowledge Items Connection Model

3.2 Item Extraction

We adopt BERT and CRF to extract mentioned items in the given document. As Figure 6 shows, the original document is input and transformed into word vectors and phrase vectors as the input of BERT. Then, we can obtain the rank of each word. FInally we use CRF to possiblity of unvalid sequence.

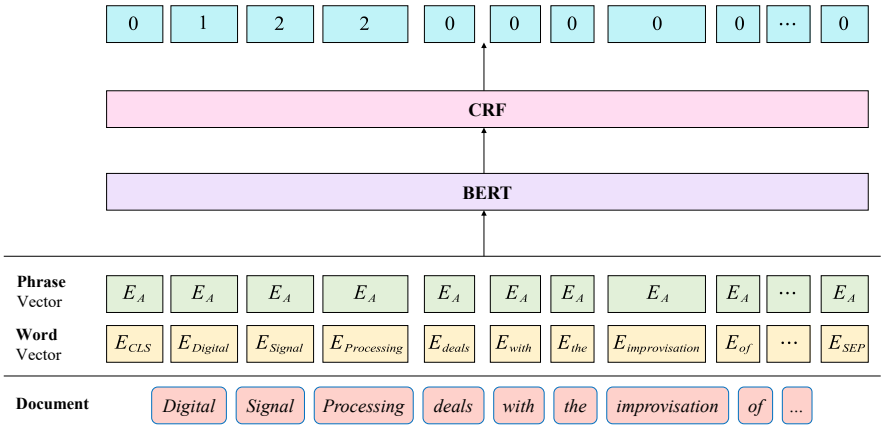

Figure 6: An example for item extraction model

The simplified process of calculating sequence labeling (SL) is as following:

$$input_{bert} = [Word, Phrase]$$

$$output_{bert} = BERT(input_{bert})$$
$$SL = CRF(output_{bert})$$

3.3 Mathematical Item Identification

Since BERT and CRF do not perform well in identifying mathematical items and figures, we choose to use Mathpix to help the system identify and extract mathematical items. Mathpix can transform the mathematical items into srings in LaTeX grammer by scanning the figure of the mathematical items, then the strings can be identified by computers[8].

3.4 Items Joint Encoding

We adopt BERT and Bi-LSTM to joint code the mentioned items and the candidate items.The model structure and its input example are shown in Figure 7.

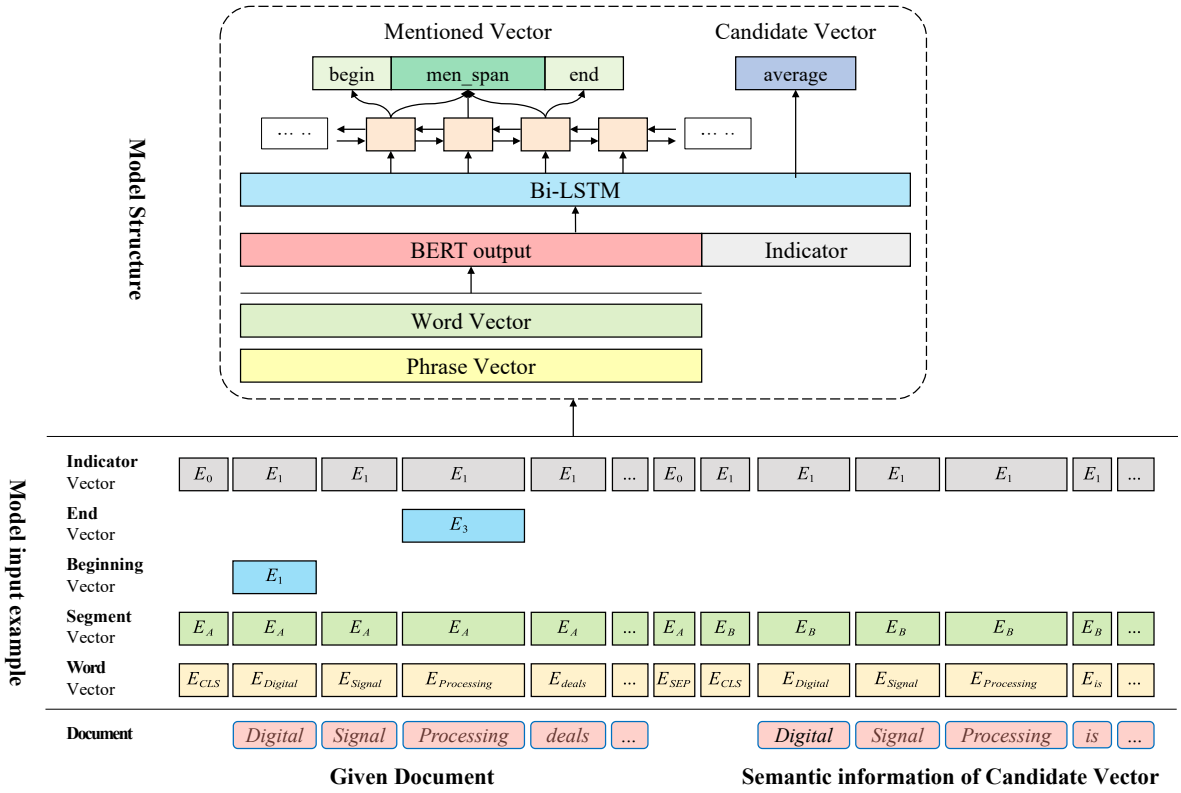

Figure 7: Model structure (upper part) and an example for input (lower part)

The input are the mentioned document and the candidate items' semantic information. After processing the input, we obtain the origin input of BERT. Apart from word vector and phrase vector,

we added beginning vector and end vector to mark the boundary of mentioned items. Moreover, a binary vector, indicator vector, is improted to indicate the importance distribution of mentioned vector and candidate vector. The simplified process of calculationg mentioned vector (MV) and candidate vector (CV) is shown as following:

$$output_{bert} = BERT([Word, Phrase])$$

$$input_{bi-lstm} = Concate([Indicator, output_{bert}])$$

$$output_{bi-lstm} = Bi - LSTM(input_{bi-lstm})$$

$$MV = Link([output^{bi-lstm}_{beginning}, \sum_{i=beginning}^{end} output^{bi-lstm}_{i}, output^{bi-lstm}_{end}])$$

$$CV = Average(output_{bi-lstm})$$

3.5 Model Output

The whole model's input and output can be expressed as:

$$output = KnowledgeItemConnection \begin{bmatrix} Word_{MV}, Phrase_{MV}, Word_{MV\&CV} \\ Phrase_{MV\&CV}, Beginning, End, Indicator \end{bmatrix}$$

$Word_{MV}$ represents the word vector of the given document, $Phrase_{MV}$ represents the phrase vector of the given document, $Word_{MV\&CV}$ represents the word vector of given document and semantic information of candidate items, $Phrase_{MV\&CV}$ represents the phrase vector of given document and semantic information of candidate items, $Beginning$ represents the beginning position vector of mentioned vectors, End represents the end position vector of mentioned vector, $Indicator$ represents the importance contribution indecate vector, and $output$ is the final consequence.

4 Two-staged Top-*k* Query Model

4.1 Algorithm Solution Framework

This chapter proposes a Two-staged Top-k Query Model for Knowledge Graph System which is not complete and has many noisy data. Firstly, the similarity model of query graph and result graph is established based on ∞-neighborhood vectorial idea. The model describes the topological relationship between a vertex and its neighbor vertex in the form of a vector and measures its similarity by the topological features of the query graph and result graph. Then, the basic idea of solving the model is proposed.

The main idea of this heuristic method is to select high-quality candidate results only for each vertex to refine the candidate set of the vertex to be searched, and generate high-quality embedding by combining high-quality candidate points, so as to achieve the purpose of greatly reducing the search

space. To achieve this, we designed a metrix to evaluate the merits of the candidate vertices, which is shown as below:

$$C(f) = \underbrace{\sum_{q_i \in V_Q} \sum_{q_j \in V_Q^S} \delta_{q_i,q_j}\left(f(q_i), f(q_j)\right)}_{Part1} + \underbrace{\sum_{q_i \in V_Q} \sum_{q_j \in V_Q^U} \delta_{q_i,q_j}\left(f(q_i), f(q_j)\right)}_{Part2}$$

In the first part, for each query vertex, we consider the relationships between the matched vertex and the matched vertex of the definite vertex whose label is known in the query graph. In the second part, we consider the relationships between matching vertices with the vertices to be looked up (that is, the vertices of the label to be found in the query graph). Since the matching vertices of definite vertices are determined and unique, high-quality candidate vertices can be selected to optimize the cost of the first part as much as possible, which can be simplified into equation as following:

$$C^{\kappa}(v, q) = \sum_{q^s \in V_Q^S} \delta_{q^s,q}\left(\phi(q^s), v\right)$$

Based on the above analysis, the Top-k Subgraph Query Algorithm is divided into two stages, and the algorithm is shown as Figure 8. In the first stage, a candidate vertex is selected for each vertex to be searched, which is a predefined constant. In the second stage, the candidate vertices of each vertex to be searched are combined to obtain the set of candidate embeddings, and eventually the optimal k embeddings are selected from the set of candidate embeddings.

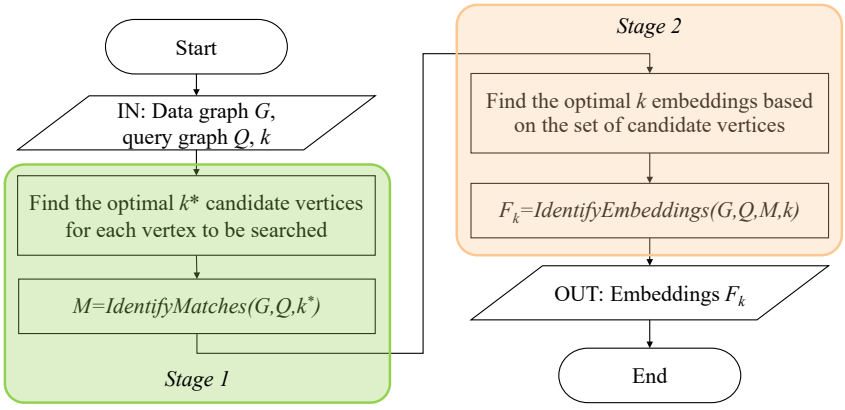

Figure 8: Complete structure of the two-staged algorithm

4.2 Detailed Algorithm Description

4.2.1 Stage One: Find the Vertex of Top-k^*

Given an item point q, any vertex of the same type as Q in the data graph can be used as a candidate vertex of Q. To find the optimal k^* candidate vertices, the most intuitive method is to first order all candidate vertices according to the candidate cost C^k, and then extract the optimal K^* candidate vertices. However, this method needs to calculate the candidate cost for each candidate vertex, which is difficult to complete quickly due to high time complexity. Therefore, we propose Threshold Algorithm (TA) Based on BFS Dynamic Update to speed up retrieval.

Based on the BFS, we dynamically generate sorted lists in real-time. According to the definition of the correlation score, the shorter the length of the shortest path between two points is, the higher the correlation score is. For a given dimension, the BFS can be used to preferentially find the candidate vertices closest to the anchor point, which will have the lowest cost.

$$\varphi_G(u,v) = \begin{cases} 1 & u = v \\ \min\left\{n_{u,v}\alpha^{l_{u,v}}, N\alpha^{l_{u,v}}\right\} & otherwise \end{cases}$$

The algorithm is shown as Algorithm 1.

Algorithm 1 *IdentifyMatches*(G, Q, K^*)

Require: G: data graph; Q: query graph; k^*: number of candidates
Ensure: optimal k^* candidates of each item point

```
//N(h, v) is the set of candidate vertices that have the distance h with vertex v
//M(q)is the Top-k* candidate set of vertex q
for each q in V_Q^U do
    for h ← 1, 2, ··· L do
        S ← ⋃_{q^s ∈ V_Q^S} N(h, φ(q^s))
        M(q) ← update(M(q), S);
        T ← Σ_{q^s ∈ V_Q^S} Θ(φ_Q(q, q^s), α^h);
        if ∀v ∈ M(q), C^K(v, q) ≤ T then
            break;
        end if
    end for
end for
return M(q_1), M(q_2), ··· , M(q_{|V_Q^U|});
```

4.2.2 Stage Two: Find the Top-k Graph Embedding

In stage two, candidate vertices of all vertices to be searched are combined to generate a candidate graph embedding set, and Top-k graph embedding is selected from candidate graph embedding. Since each vertex to be searched has K^* candidate vertices, the number of candidate graph inserts

is K. When k or V is large, traversing the candidate graph embedding set will be a time-consuming operation. This section introduces the idea of Branch Bound to speed up the whole traversal process.

In this paper, the expectation of vertex approximation costs evaluates the pros and cons of V instead of the exact vertex approximation costs, which is shown as following:

$$\bar{c}\left(v, q_i\right) = C^K\left(v, q_i\right) + \sum_{q_j \in V_Q^U} \frac{\sum_{u \in M\left(q_j\right)} \delta_{q_i, q_j}\left(v, u\right)}{\left|M\left(q_j\right)\right|}$$

$C^K\left(v, q_i\right)$ is the candidate cost of vertex V, and $\sum_{q_j \in V_Q^U} \frac{\sum_{u \in M\left(q_j\right)} \delta_{q_i, q_j}(v, u)}{\left|M\left(q_j\right)\right|}$ represents the mean value of correlation between V and candidate vertices.

To get the difference of candidate vertices, the variance of candidate vertices is calculated first, and then the vertices to be searched are sorted from large to small according to the variance of candidate vertices. The vertices to be searched with a large variance will correspond to the layer closer to the root.

The algorithm is shown as Algorithm 2.

Algorithm 2 *IdentifyEmbeddings*(G, Q, M, K)

Require: G: data graph; Q: query graph; M: candidate set; k^*: number of candidates
Ensure: optimal k embedding graphs

$List_Q \leftarrow RankQueryNode(V_Q)$;
$Frontier \leftarrow \{\tau_{root}\}$;
$F_k \leftarrow varnothing$;
while $Frontierisnotempty$ **do**
  $\tau \leftarrow Frontier.getBest()$;
  if $\check{C} \geq \lambda$ **then**
    break;
  end if
  $\{\tau_1, \tau_2, \cdots\} \leftarrow \tau.nextLevel(List_Q)$;
  if $\{\tau_1, \tau_2, \cdots\}\, areleaf - nodes$ **then**
    $F_k \leftarrow UpdateTopkSet(F_k, \{\tau_1, \tau_2, \cdots\})$;
    $\lambda \leftarrow GetThreschold(F_k)$;
  else
    $Frontier \leftarrow Frontier.add(\{\tau_1, \tau_2, \cdots\})$;
  end if
end while
return F_k;

4.3 Model Evaluation

To evaluate the performance of our model, four creterions are given: Subgraph Isomorphism, Approximate Matching, Data Organazation, and Query Speed. Then we compare oue model with existing algorithms including RDF-3X[10], Ness[11] and NeMa[12], the effectiveness and performance of the algorithm are analyzed.

Algorithm	Subgraph Isomorphism	Approximate Matching	Data Oraganization	Query Speed
RDF-3X	Supported	Nonsupported	Necessary	Fast
Ness	Supported	Partially Supported	Unnecessary	Slow
NeMa	Nonsupported	Partially Supported	Unnecessray	Fast
Ours	Supported	Supported	Unnecessary	Fast

Table 2: Camparing with present algorithms

5 Self-adaption Algorithm Based on Multiple Potential Vectors

To improve a knowledge graph system's content and structure, we believe that potential knowledge, which may exists in the knowledge graph, and knowledge triples including new pieces of knowledge should be predicted. Previous research has had the ability to predict potential knowledge by using existing knowledge. However, due to lack of detailed semantic information of the new pieces of knowledge, to predict knowledge triples becomes more challenging. Thus, inspired by the Zero-shot Learning Algorithm[9], we design the Self-adaption Algorithm based on multiple potential vectors.

To calculate the potential semantic eigenvector of items by using their semantic information, we design the Potential Semantic Vector Learing Algorithm based on the semantic information of items. Based on this algorithm and the knowledge graph, we design the Multiple Potential Vectors Model.

5.1 Potential Semantic Vector Learning Algorithm

We suppose that there exist two items A and B. Since their semantic information is formal specifications, it can be expressed as $d_e := \{w_1, w_2, \ldots, w_n\}$, in which d_e represents the semantic information of item e, n represents the size of the set of words, and $\{w_1, w_2, \ldots, w_n\} \in W$ represents the set of words in the semantic information without stop words. We use $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n \in \mathbf{W}$ to indicate the potential semantic vectors, namely word vectors. By using Weighted Bag of Words (WBOW) to model the semantic indormation, then we can calculate the weights of words.

First, calculate the frequency ratio of w_i in the semantic information of items d_e:

$$F(w_i, d_e) = \frac{f(w_i, d_e)}{\sum_{w \in d_e} f(w, d_e)}$$

$f(w, d_e)$ indicates the frequency of occurrence of word w in the semantic information d_e.

Then, calculate the inverse frequency of w_i in the semantic information of all items D:

$$IF(w_i, D) = \ln \frac{|D|}{|\{d_e \in D : w_i \in d_e\}|}$$

D is the set of semantic information, $|D|$ indicates the amount of semantic information in D, and $|\{d_e \in D : w_i \in d_e\}|$ represents the amount of semantic information including the word w_i.

After that, the weight of w_i in d_e can be calculated:

$$weight(w_i, d_e, D) = \frac{F(w_i, d_e) \times IF(w_i, D)}{\sum_{i=1}^{n} F(w_i, d_e) \times IF(w_i, D)}$$

and $\sum_{i=1}^{n} weight_i = 1$.

Eventually, the potential vector of item e is:

$$\mathbf{e}_d = \sum_{i=1}^{n} weight_i \times \mathbf{w}_i$$

$weight_i \in (0, 1)$, $\mathbf{w}_i \in \mathbb{R}^\kappa$, in which κ is the dimension of word vectors.

5.2 Multiple Potential Vectors Model

To calculate the word vector in the semantic information by using the structal knowledge in thr knowledge graph system, we define a knowedge triple as $< subject(s), predicate(p), object(o) >$. Its function $G(s, p, o)$ is:

$$G(s, p, o) = f_{sp}(s, p) + f_{po}(p, o) + f_{so}(s, o)$$

$f_{sp}(s, p)$, $f_{po}(p, o)$, and $f_{so}(s, o)$ are the interactions between two elements in the triple. The interactions mean the co-occurrence of two elements. We believe that two elements' potential semantic vectors are close when their co-occurrence is much. Therefore, the stronger three interactions are, the bigger the function value is, which indicates the triple has higher possibility to be valid. Moreover, to distinguish the direction of interactions, the interactions are expressed as two potential vectors (p_s, p_o), which are corresponding with the subject and object.

Then, we can calculate the potential vectors of subject and object through algorithm in section 4.1:

$$s_d = \sum_{i=1}^{n} weight_{si} \times \mathbf{w}_{si}, o_d = \sum_{i=1}^{m} weight_{oi} \times \mathbf{w}_{oi}$$

n and m are the sizes of the word set in subject and object.

Thus, we can construct the Multiple Potential Vectors Model on the foundation of equations above,

its fuction is:

$$\begin{aligned} G(s,p,o) &= f_{sp}(s,p) + f_{po}(p,o) + f_{so}(s,o) \\ &= s_d \cdot p_s + p_o \cdot o_d + s_d \cdot o_d \\ &= (\sum_{i=1}^{n} weight_{si} \times \mathbf{w}_{si}) \cdot p_s + p_o \cdot (\sum_{i=1}^{m} weight_{oi} \times \mathbf{w}_{oi}) \\ &\quad + (\sum_{i=1}^{n} weight_{si} \times \mathbf{w}_{si}) \cdot (\sum_{i=1}^{m} weight_{oi} \times \mathbf{w}_{oi}) \end{aligned}$$

Thus, for a new item with semantic information, we can first calculate its potential eigenvector, and then predict the knowledge triple including the new item. For example, when we want to predict the object in a triple:

$$\mathbf{TopN}\,(s,p,?) := \arg\max_{\hat{o}\in\hat{\mathscr{E}}}^{N}\ \mathbf{G}\,(s,p,\hat{o})$$

N is the amount of alternative predicted objects. Similarly, we can use the same method to predict the subject in a triple.

6 Visualized Software Design

6.1 Architecture Description

To present our knowledge graph system, we design a visualized software. This software adopts B/S (Browser/Server) as architecture, and consists of front-end browser and back-end server. The browser is used to interact with users and visualize knowledge while the server is used to implement functions and store data. Considering the functions in previous sections, the layered architecture is adopted to the software. The architecture is shown as Figure 9. There are four layers: Presentation Layer, Interface Layer, Service Layer, and Persistence Layer.

1. Presentation Layer: used to visualize the relations between items, interact between users and software. For example, item query, relation query, label edit, relation edit, etc.

2. Interface Layer: the intermediation between Presentation Layer and Service Layer. It procides uniform functional interfaces for Presentation Layer, which efficiently improve the expansibility of software. Based on the demand of Presentation Layer, Interface Layer mainly provide Presentation Layer with item query interfaces and knowledge update interfaces.

3. Service Layer: the core of software. It is used to complement functions and logic and provide functional service for upper layers. For example, item query service, relation query service.

4. Persistence Layer: the data strorage layer. It provides persistence service for upper layers, and used to retrieve and store data from and in the database.

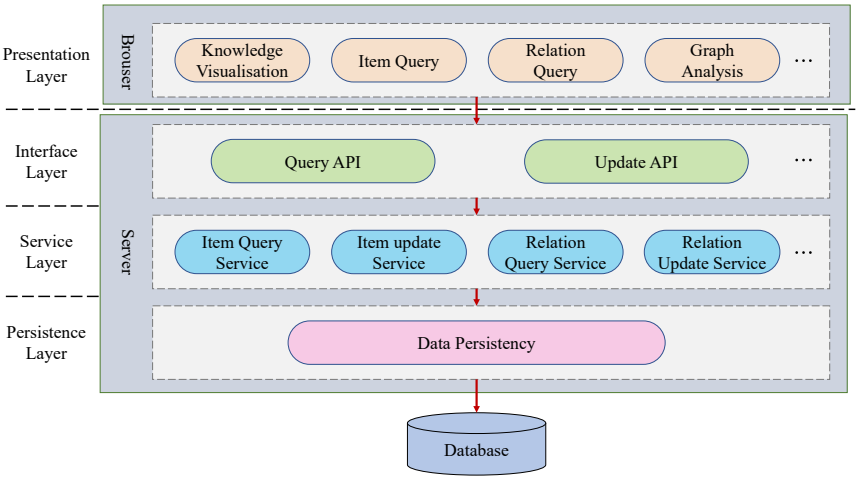

Figure 9: Architecture of software

6.2 Back-end Database

We use Neo4j to be our graph database because it support Python, has comprehensive function and graph algorithm, and visualization system. Apart from that, since our database is relatively small, Neo4j can meet our need.

6.2.1 Item Query

Item query is the core function of our software, which is aimed to efficiently query and retrieve items. According to the request from the front-end, the back-end server request for querying Neo4j database and return the data correspongding to the query items. Finally, the item being queried is visualized through front-end browser.

For example, when user input "AWGN_channel" and query, the outcome is shown as Figure 10. It shows the associated items and relations of "AWGN_channel". We can see from the figure that the mathematical item $r(t) = s(t)+n(t)$ is the expression formula of AWGN_channel and Selfcorrelation is the property of AWGN_channel.

6.2.2 Relation Query

Relation query is aimed to help users search for the relation between two items. There are three input mode of users in relation query service:

1. User inputs one item: similar with item query, users can query all items and relations associated with input item.

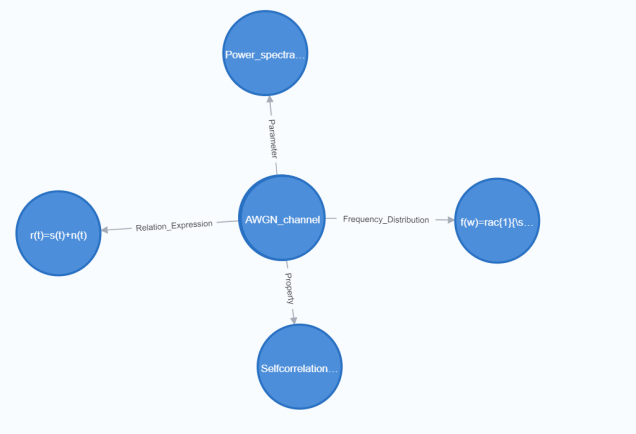

Figure 10: Example for item query service

2. User inputs two items: the relation between the input items can be straightly query. For example, the user input "AWGN_channel $r(t) = s(t) + n(t)$" and query, then the outcome is shown as Figure 11. From the figure, we can see that the system gives that $r(t) = s(t) + n(t)$ is the expression formula of AWGN_channel.

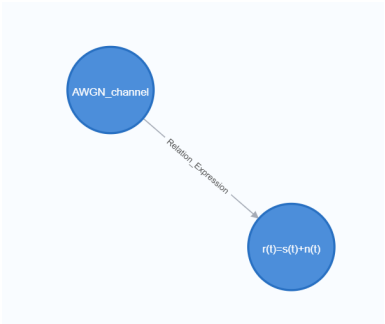

Figure 11: Example for input mode two

3. User inputs one item and one type of relations: all items satisfying the relation with the input item will be queried. For example, the user input "AWGN_channel Frequency Distribution" and query, then the outcome is shown as Figure 12. From the figure, we can see that the system gives the mathematical item $f(w) = frac1\sqrt{2\pi}\sigma \exp(-frac(x-\mu)^2 2\sigma^2)$, which is the frequency distribution of AWGN_channel.

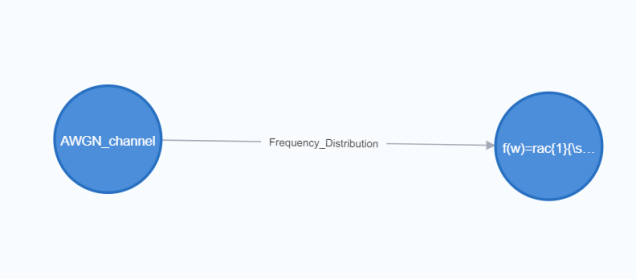

Figure 12: Example for input mode three

6.3 Simplified Presentation

The simplified presentation of our software and knowledge graph system is shown as Figure 13.

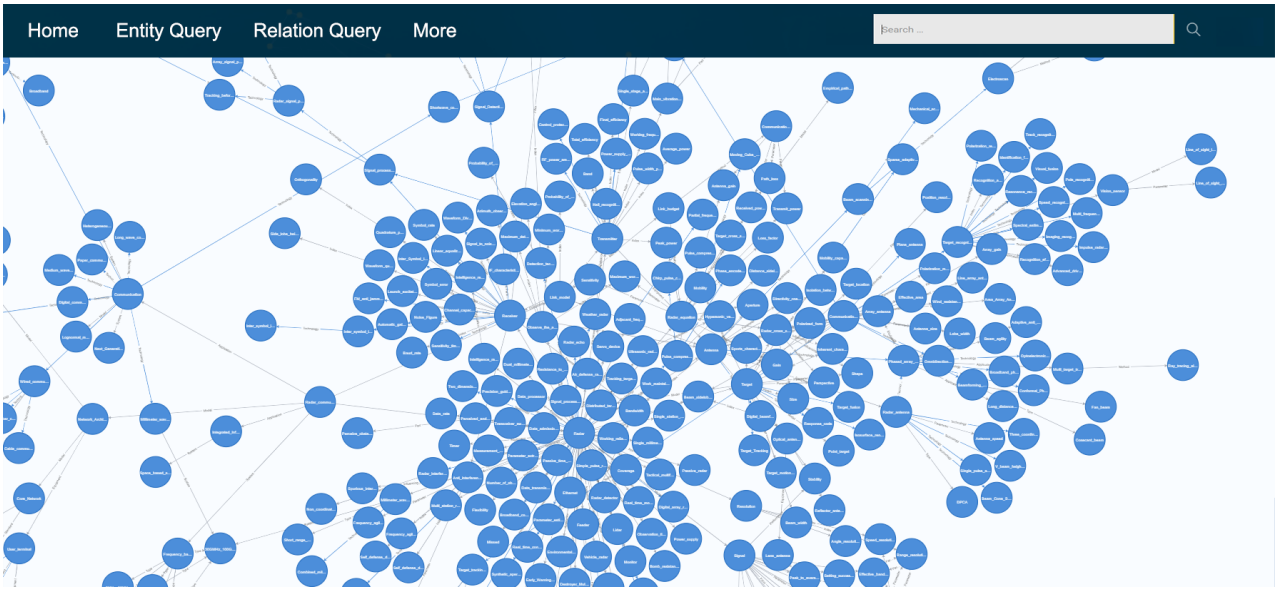

Figure 13: Simplified presentation

7 Conclusion

We have constructed a self-adaptive Knowledge Graph System and its visualized software. First, we extracted the items and labels within communication, reconnaissance, detection and interference. Then, the subclass-of and instance-of relations were connected by using NLIP. After that, a System Evaluation Model was built and three tyoes of quality dimensions were uesd to evaluate the performance of our system.

For the second question, we gave four rules to connect candidate items to new items with semantic information and built a Knowledge Items Connection Model based on joint encoding, namely made the system have the ability to query fuzzy items.

About the query under restricted conditions, we built a Two-staged Top-k Query Model. It performed very well on each of the four creterions we gave.

Next, we designed a mechine learning algorithm called Self-adaption Algorithm to help our system have the ability to self-apapt, and the algorithm is based on multiple potential vectors.

Finally, we designed a visualized software with layer architecture. The software successfully implemented functions above and visualized our Knowledge Graph System.

References

[1] L. Ehrlinger and W. Wob, "Towards a Definition of Knowledge Graphs." Institute for Application Oriented Knowledge Processing Johannes Kepler University Linz, Austria. [Online]. Availible: http://ceur-ws.org/Vol-1695/paper4.pdf

[2] Q. Liu, "Knowledge Graph Construction Techniques," *Journal of Computer Research and Development*, vol. 53, no. 3, p. 582-600, 2016, doi:10.7544/issn1000-1239.2016.20148228.

[3] Wikipedia, "Knowledge Graph". [Online]. Availible: https://en.wikipedia.org/wiki/Knowledge_graph

[4] E. Feng, "Research on the Construction Method of Knowledge Graph for the Integration of Detection and Communication," University of Electronic Science and Technology of China, Chengdu, 2021.

[5] R. Lou, "Taxonomy Induction Research on Knowledge Base from Chinses Encyclopedia," Zhejing University, Hangzhou. Accessed: Jan. 2016.

[6] A. Zaveri, A. Rula,"Quality Assessment Methodologies for Linked Open Data," Accessed: Jun. 2016. [Online]. Availible: http://www.researchgate.net/publication/235912899_Quality_Assessment_Methodologies_for_Linked_Open_Data

[7] R. Delbru, N. Toupikov, "Hierarchical link analysis for ranking web data," *The Semantic Web: Research and Application*, Berlin, 2010: pp. 225-239. doi: 10.1007/978-3-642-13489-0_16.

[8] F. Li, "Extraction, Recognition and Reconstruction of Mathematics Formulas in English Scientific Document," Dalian University of Technology, Dalian, 2007.

[9] Y. Zhao, "Research on Knowledge Graph Automatic Evolution," Beijing University of Posts and Telecommunications, Beijing. Accessed: Sept. 2017.

[10] T. Neumann, G. Weikum, "RDF-3X: A RISC-style Engine for RDF," *Proc. VLDB Endow*. 2008: vol. 1, no. 1, pp. 647-659.

[11] A. Khan, N. Li, "Neighborhood Based Fast Graph Search in Large Networls," in Procceedings of the 2011 ACM SIGMOD International Conference on Management of Data (SIGMOID'11), 2011: pp. 901-912.

[12] A. Khan, Y. Wu, "NeMa: Fast Graph Search with Label Similarity," *Proc. VLDB Endow*. 2013: vol. 6, no. 3, pp. 181-192.

[13] N. Noy. and D. McGuinness, "Ontology development 101: A guide to creating your first ontology." Accessed: Dec. 2010. [Online]. Availible: http://www.sfu.ca/dmarques/federation/pdf/Noy-Ontology101.pdf.2001.

[14] W. Fang, "Research on Semantic Retrieval for Communication Ontology," Jiangxi Normal University, Nanchang. Accessed: Jun. 2015. [Online]. Availible: https://kns.cnki.net/kcms/detail/detail.aspx?FileName=1015624709.nh&DbName=CMFD2016

[15] T. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. 5, (2), pp. 199-220, 1993.