Design and Implementation of an Academic Search System Based on a General Query Language and Automatic Question Answering

Zi Xiong[1], Yue Qi[1], Wei Lu[1], Qikai Cheng[1]
[1] School of Information Management, Wuhan University, Wuhan, China
zixiong@whu.edu.cn,yueq2017@gmail.com,weilu@whu.edu.cn,chengqikai0806@163.com

ABSTRACT

This research designs and implements an academic search system with two major innovations: 1) proposing a general query language SSL to describe the academic search intention in a unified and standardized way, 2) proposing a user intention recognition method to help to improve traditional automatic question answering systems, and conduct several experiments to prove the effectiveness of our method. We finally applied the SSL language and intention recognition method to a QA-oriented academic system which is innovative compared with traditional query-based systems.

KEYWORDS

Academic Search; Search intention; Domain Specific Language; Task-oriented Dialogue System

1 INTRODUCTION

The academic search engine is a key tool for scholars to access academic information. Search engines such as Google scholar and Baidu Xueshu have played an important role in researchers' work. These engines index millions of scholarly documents but overwhelm their users with hundreds of results in response to a simple query. Nowadays, automatic question answering(QA) has achieved big success in mobile search and helps to improve the efficiency of search engines[1, 2]. In this paper, we design and implement a general query language for both academic search and academic QA. As an applied validation challenge, we build a QA-oriented academic search engine that achieves good performance in a real application.

Our work focuses on solving two problems: *design of a general query language for academic search* (Section 3) and *understanding the search intentions of users' questions* (Section 4). We further validate the solutions of these two problems with a *implemented system of QA-oriented academic search engine* (Section 5). Our complete research framework is shown in Figure 1.

2 SSL: SCHOLAR SPECIFIC LANGUAGE

We propose a scholar specific language **SSL** for both academic search and academic QA. The SSL designed in our work is a semantic representation of the user's search intention that can help different academic search engine implementation understand users' search intentions in a unified way. A Search/QA system is usually composed of two key modules: query understanding module (QUM) and information processing module (IPM). In most of the cases, these two modules tightly coupled. With the introduction of SSL designed in our research, QUM and IPM can be decoupled. We refer to the traditional design process of domain-specific languages[3, 4]

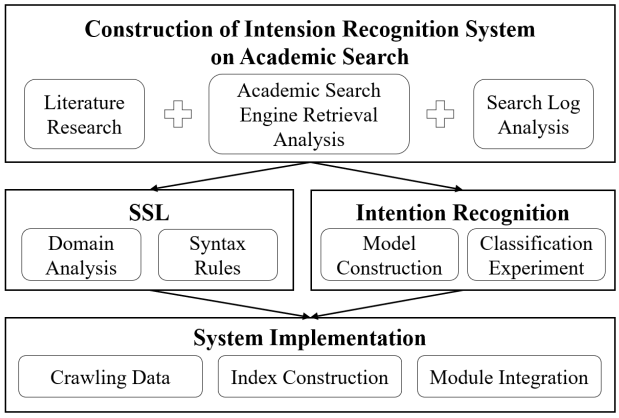

Figure 1: Research Framework.

when defining **SSL**. In semantic, we define the classification and composition of academic search intentions based on the analysis of search log analysis and academic search engines. In grammar, we describe the corresponding grammatical rules based on JSON format through ABNF[5].

The details of the SSL definition is presented in appendix A. The grammar of SSL consists of 20 rules with the left side of "=" as the name of the rule and the right side as the body of the rule. The body consists of decomposed elements with "," as connectors, which may be subrules or terminal symbols such as "", "", etc. In general structure, SSL mainly consists of 3 modules:

1. Type module, which is used to represent the type of purpose information or intention.

2. Field module, which expresses the query mode which is the specific combination of destination information attributes.

3. Refinement module, which is used to represent the refined query semantics of the result information, includes both the post-filtering semantics and the secondary retrieval semantics.

3 UNDERSTANDING USER INTENTION

3.1 Intention Recognition

The functional modules of our task-oriented dialogue system include question understanding, dialogue management, and search modules. Our research focuses on the intention recognition task[6, 7], which aims to improve the understanding of the user's search intention. We use text classifiers to infer user's search intention through dialog text and conduct intention recognition experiments on several real-world datasets, and prove that the model based on RNN classifier has the best effect of all the classifiers.

After the classification model, a few steps are then taken to implement our intention function.

- After classification, we define several question templates to match the search method.
- In dialog management, a strategy based on the finite state is adopted for state tracking, and a dialog is generated based on the knowledge base and templates.
- Before retrieval, the SSL mapping and parsing module can replace the user intentions by SSL expressions.

The detail of the dialog system is shown in Figure 2. With the dialog system design, our method can provide a reusable framework for academic search to map user input to general query language.

3.2 Experiment

Dataset and Labeling. We use the search log of *Baidu Academic* and *Luojia Academic search* as the data of intention recognition experiment. The dataset are labeled with 4 query intention labels: literature query, academic entities query, academic concept query, and free question-answering. It includes 20000 query texts automatically labeled based on rules, word segmentation, and entity recognition to enrich the data, and 1500 manmually labeled. 1000 of the 1500 manmually labeled data are used for training and the other 500 for testing, ensuring the result is reliable.

Table 1: Comparison with Baselines.

Methods	Precision_{macro}	Recall_{macro}	F1_{macro}
Naive Bayes	0.6969	0.6952	0.6855
Logistic Regression	0.9506	0.9549	0.9505
SVM	0.9136	0.9124	0.9126
MV-LSTM(Ours)	**0.9519**	**0.9512**	**0.9513**

Result and Analysis. Based on the above datasets, search intention categories are divided into four major classes *literature*, *academic entity class*, *academic concept* and *free question-and-answer*. The experiment compares four classification algorithms (Naive Bayes, SVM, Logistic regression, and MV-LSTM[8]), among which Naive Bayes is taken as the baseline. The experiment result is shown in Table 1. Note that we don't use complicated language models like BERT because we aim to develop a lightweight framework that is universally available in real applications.

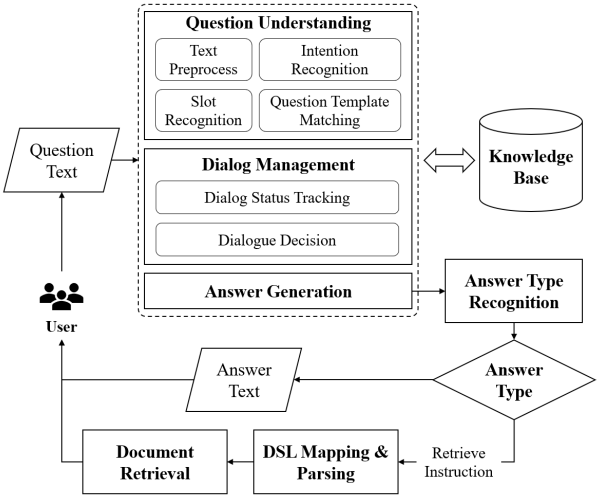

Figure 2: Dialog System Framework.

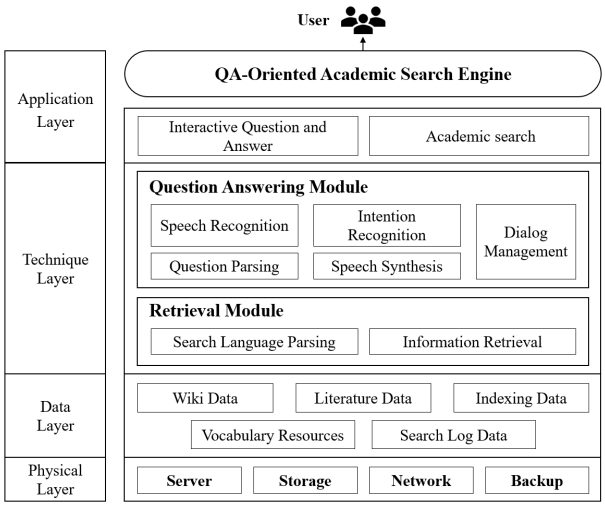

Figure 3: System Framework.

The result shows that the MV-LSTM model achieves the best performance in our intention classification task and outperforms the baseline by 26.6%, proving its usefulness.

4 QA-ORIENTED ACADEMIC SEARCH ENGINE

4.1 System Framework

The system architecture can be logically divided into the physical layer, data layer, technical layer, and application layer. The physical layer briefly describes the construction of the underlying server and storage device. The data layer describes the storage of structured and unstructured data. The technical layer mainly involves two core modules of the system question answering module and retrieval module. The application layer describes that the system provides interactive QA-oriented academic search services with a web-based engine. The detailed system framework is shown in Figure 3.

4.2 System Deployment

As shown in Figure 4, there is a dialog box on the left for users to conduct QA-oriented information retrieval(results shown in the bottom-right list). Users can also get answers directly through conversation with QA module. Our system also provides a set of retrieval tools on the upper-right for users to conduct traditional academic retrieval. Our system has been integreted into Luojia academic search engine *Xiaobu*.

Figure 4: System Interface.

REFERENCES

[1] Amit Mishra and Sanjay Kumar Jain. A survey on question answering systems with classification. *Journal of King Saud University-Computer and Information Sciences*, 28(3):345–361, 2016.

[2] Yang Li, Qingliang Miao, Ji Geng, Christoph Alt, Robert Schwarzenberg, Leonhard Hennig, Changjian Hu, and Feiyu Xu. Question answering for technical customer support. In *NLPCC*, pages 3–15. Springer, 2018.

[3] Martin Fowler. *Domain-specific languages*. Pearson Education, 2010.

[4] Diomidis Spinellis. Notable design patterns for domain-specific languages. *Journal of systems and software*, 56(1):91–99, 2001.

[5] Dave Crocker and Paul Overell. Augmented bnf for syntax specifications: Abnf. Technical report, RFC 2234, November, 1997.

[6] Broder and Andrei. A taxonomy of web search. *Acm Sigir Forum*, 36(2):3, 2002.

[7] Madian Khabsa, Zhaohui Wu, and C Lee Giles. Towards better understanding of academic search. In *2016 IEEE/ACM Joint Conference on Digital Libraries (JCDL)*, pages 111–114. IEEE, 2016.

[8] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. A deep architecture for semantic matching with multiple positional sentence representations. In *AAAI*, volume 16, pages 2835–2841, 2016.

A DEFINITION OF SSL

- intent = "" module ""
- module = type "," field ["," refinement]
- type = DQUOTE "type" DQUOTE ":" intent-category
- field = DQUOTE "field" DQUOTE ":" fields
- refinement = DQUOTE "refinement" DQUOTE ":" refinements
- intent-category = DQUOTE ("paper"/"citation"/"entity"/"concept"/"qa") DQUOTE
- name = "subject", "author", "time", "title", "keywords", "abstract", "source", "institution", "foundation", "doi", "classification_code", "content", "paper_type", "journal", "conference", "question", "concept"
- item = DQUOTE [bool] 1*char DQUOTE
- bool = "+" / "|" / "-"
- refinements = "" quantity "," rank ["," field] ""
- quantity = DQUOTE "quantity" DQUOTE ":" quantity-item
- quantity-item = "-1" / number
- rank = DQUOTE "rank" DQUOTE ":" rank-item
- rank-item = DQUOTE ("relevence" / "citations" / "download_num" / "time") DQUOTE
- number = D *DIGIT
- D = "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9"
- char = unescaped / escaped
- unescaped = %x20-21 / %x23-5B / %x5D-FF
- escaped = %x5C ("" / "̈" "b" / "f" / "n" / "r" / "t" / ("u"4(ALPHA / DIGIT)))