DESIGN AND IMPLEMENTATION OF THE ANNOTATION MODULE IN COLDS

BY

XIN WEI

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Adviser:

Professor Chengxiang Zhai

ABSTRACT

This thesis describes the design and implementation of the annotation module in COLDS (Cloud-based Open Lab for Data Science) system. COLDS is a general infrastructure system to support data science programming assignments on the cloud that is currently being developed at the University of Illinois at Urbana-Champaign. The annotation module served as a key module among all the modules the COLDS system may have. The annotation subsystem of COLDS is responsible for allowing instructors designing and distributing annotation tasks, and allowing annotators annotating searched documents. The function of the annotation module includes providing communication between search engine and database. It also supports doing relevance judgment annotation on the results returned from search engine and store them into database. The thesis describes the design and implementation of the annotation module, including the requirements, the design of data schema, the choice of data structures, important implementation details, and sample screenshots to illustrate its applications. Also, it introduces the COLDS' background, related work, sample tasks, challenges and future work.

To my girlfriend who never showed up in my Master's program.

ACKNOWLEDGMENTS

Thanks to my advisor Professor Chengxiang Zhai for giving me this great opportunity to work on such great project. Thanks to my teammate Xiaofo Yu, Chaoqun Liu, and Guoqiao Li, it is a great pleasure to work with you guys. Thanks to my neighbor Peiyuan Zhao for all the help provided. Thanks to all the staff of Daya Tech.

Thanks to my parents, my family and all my friends for all the supports, I cant finish my thesis without you.

Thanks to Yitong Li for the great performance in The Legend of the Condor Heroes 2017, you were my spiritual support on every Monday and Tuesday. Thanks to my waifu, Aragakki Yui, you were my spiritual support last semester.

Thanks to Evo, Miga, God Fella, Star Karoake, and all the restaurants in Urbana-Champaign area, without you, I would have to cook my own dinner and may not have enough time to finish my thesis.

Thanks to Netease Music, I cant finish my thesis without your music. Thanks to BiliBili, AcFun, hupu and zhihu, without you I would have finished my thesis a month ago. Thanks to Blizzard Entertainment, you weaken Genji too much that I almost give up on Overwatch, otherwise I wont be able to finish my thesis.

Thanks to Amazon and Microsoft for give me fulltime offer so I can focus on my thesis without worrying be unemployed. Thanks to Bloomberg and Goolge for give me onsite interview, though you didnt give me an offer in the end.

At last, thanks to University of Illinois at Urbana Champaign and the CS department, you made me who I am today, I will miss you so much after my graduation.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1
1.1 Background 1
1.2 Motivation 2
1.3 Annotation Module 2

CHAPTER 2 RELATED WORK 4
2.1 Big Data and Big Data Education 4
2.2 Virtual IR Lab and Lucene4IR 5
2.3 Crowdsourcing Annotations 5
2.4 Information Retrieval Evaluation 6

CHAPTER 3 COLDS AND ITS ANNOTATION SUBSYSTEM . . . 7
3.1 Overview 7
3.2 Structure of COLDS Annotation Subsystem 7
3.3 Use of Annotation Subsystem 8

CHAPTER 4 DESIGN OF ANNOTATION MODULE 9
4.1 Design Philosophy 9
4.2 Challenges and Solutions 9
4.3 Overall Structure 11

CHAPTER 5 IMPLEMENTATION DETAIL 13
5.1 Flask Server with RESTful API 13
5.2 Bootstrap with jQuery Front End 13
5.3 MongoDB and Mongoengine 14

CHAPTER 6 APPLICATIONS OF THE COLDS' ANNOTATION SUBSYSTEM 15
6.1 Education Application 15
6.2 Benefits of COLDS' Annotation Subsystem 16

CHAPTER 7 SUMMARY 18

REFERENCES 19

CHAPTER 1

INTRODUCTION

This chapter describes the background, motivation and brief overview of COLDS and its annotation module.

1.1 Background

The growth of the "big data" created massive opportunities in the industry. Many companies hire specific data scientists roles to apply big data techniques such as data mining, information retrieval and other machine learning algorithms to find patterns in the massive raw data for better advertising or user behavior analyzing which can improve their productivity and make more profits. Such trend has caused a big gap in the big data industry for engineers who know big data techniques. As a result, there is great demand for training a large number of data scientists and engineers quickly and at a low cost.

Massive Open Online Courses(MOOC) education platforms such as Courser, Udacity and EdX have the technique to support education at a very large scale. However, the way those MOOC platforms running is more like online classrooms where students can watch lecture videos and working on online quizzes. Some of the courses also requires peer grading for complex assignments at a large scale, but the quality of such peer grading is questionable. But the more serious hold back of those MOOC platform is that it doesn't support programming assignments, which is very essential for computer science education, especially for data science education.

1.2 Motivation

Running data science models requires large amount of data. When doing big data education, especially online education like Coursera and Edx, keeping the copy of the datasets locally on students' machine can be a waste of resource. Also, when evaluating different data science models, algorithms and rankers, the difference between OS, hardware, and setups can have multiple affections on the results. Thus, a cloud based data science platform can solve the above issues in an efficient way. With the help of cloud platform, users can reuse the large dataset which is stored on the cloud. The cloud platform can also provide a uniform environment, including OS, hardware and configurations to run and evaluate data science models, algorithms and rankers.

To address this challenge, the Text Information Management and Analysis group at the University of Illinois at Urbana-Champaign has been developing a novel Cloud-based Open Lab for Data Science (COLDS) [1] to enable students to work on programming assignments involving big data sets on the cloud.

COLDS is based on the powerful MeTA toolkit[1]. The toolkit can support multiple data science models and algorithms. It is an open-source state-of-art Information Retrieval toolkit. With the help of MeTA toolkit, COLDS can provide good and stable performance in the cloud. Since the toolkit is open-source, users can justify and moderate the toolkit according to their needs.

On behalf of all the characteristics of the COLDS, it is a great tool for Big Data education and run controlled data science experiments.

1.3 Annotation Module

A novel and important subsystem of COLDS is its Annotation Subsystem, which not only enables students to work on data annotation assignments on the cloud, but also facilitates creation of new annotated data sets to support novel research in data science.

This thesis describes the design and implementation of the annotation

[1] https://wiki.illinois.edu/wiki/pages/viewpage.action?pageId=586660414

module of the Annotation Subsystem of COLDs. The whole Annotation Subsystem consists of 3 modules, including search engine module, database module, and annotation module. The description of the design and implementation of the other modules of the Annotation Subsystem can be found in Liu[2] and Yu[3] master thesis.

Annotation module serves the key functionality of the COLDS system. It interacts with the rest of the parts as follows: when the search module returned the search engine result, it will pass the result to the annotation module. Then the annotation module will save the annotation for each document into the database module. These annotations can be further used to do evaluations or used as explicit feedback to enhance the search result. When doing big data educations, instructor can also assign the students tasks to do annotation on specific dataset with certain algorithms and data science models for educational or experimental purposes.

The goal of the annotation module is to let the instructors of big data to assign annotation tasks to students and save these annotations for future use such as run experiments against different data science models. The annotation module also needs to serve other parts. It uses the result from search engine module as the input and saves its output to the database module.

The annotation module enabled COLDS to support 1). flexible design of annotation assignments 2). experiments for evaluation of different data science models and algorithms 3). interaction between different modules. There are 2 main challenges in designing this module, design the database schema and design the data structure. We discuss how we address these challenges and present the details of the design and implementation of the annotation module in chapter 4 and chapter 5. We also show sample results of using COLDS system to illustrate its potential applications in chapter 6. The system is available at https://github.com/BtXin/VirtualIRLab

CHAPTER 2

RELATED WORK

This chapter describes related works including relevant background, similar works, and techniques behind COLDS.

2.1 Big Data and Big Data Education

The opportunity of leveraging large amounts of data (i.e., "big data") in all kinds of application domains has been recognized by the US government for several years now [4]. In the past a few years, much progress has been made in big data research as shown by the creation of new conferences dedicated to this topic such as the IEEE Big Data Conference [1]. On the education side, many universities have created new courses on Data Science; for example, a search with "data science" as a query on Coursera (https://www.coursera.org/) returns a large number of online courses in the general area of Data Science. However, a significant challenge in these online courses is how to enable students to work on meaningful large-scale programming assignments involving large data sets. There have been a few systems that can help support online programming assignments in Data Science. Some examples of them are the MLComp [5] and Kaggle[6], which support some machine learning experiments with all the experiment details documented in the system. However, the system does not support grading of programming assignments, or enable very large data sets to be used. It also does not support annotations of data sets as COLDS does.

[1] http://cci.drexel.edu/bigdata/bigdata2017/index.html

2.2 Virtual IR Lab and Lucene4IR

Virtual Information Retrieval Laboratory was a web based information retrieval laboratory[7]. It used a more interactive way of implementing information retrieval functions, unlike the pre-existing command line based toolkits. Users can run their retrieval functions directly over the data sets stored in the VIR Lab's server, which is similar to COLDS system. Users can implement the retrieval function with few lines of code, and can process evaluation and pair-wise comparison with the web interface.

However, Virtual IR Lab doesn't have an annotation system. As a result, it cannot support task assignment and direct explicit feedback. Also, the Virtual IR Lab is not open sourced, which means users can't deploy and modify the system according to their needs. Thus, Virtual IR Lab is not as suitable as COLDS for big data education.

Azzopardi described Lucene4IR[8]. in his workshop paper as an evaluation process tool. The tool used Apache Lucene toolkit, which is mostly used in industry. The tool is mainly used for academic purposes. They chose Lucene to decrease the gap between academic and industry.

However, the tool is not cloud-based, which means you have to keep the data sets locally for each user. It also doesn't have an annotation subsystem.

2.3 Crowdsourcing Annotations

Creation of annotated data sets is required in order to evaluate algorithms in Data Science. However, annotations generally require much manual labor, thus is expensive. This limited the availability of annotated data sets that can be used for supporting new research in data science. Recently, crowdsourcing annotations of data sets has become very popular due to its affordability[9] [10] [11]. However, existing methods mostly rely on paying many cheap labors (e.g., using Amazon Mechanical Turk). In contrast, the Annotation Subsystem of COLDS enables a novel way of crowdsourcing annotations of potentially very large data sets by leveraging student assignments where a large data set can be distributed among many students to create annotations and all the annotations can then be aggregated to form a large data set. This novel strategy potentially enables creation of annotated data sets without any

cost since the students would learn evaluation skills from working on such annotation assignments. Such an assignment-based annotation strategy has proven effective when used in the Text Retrieval and Search Engines course on Coursera (https://www.coursera.org/learn/text-retrieval).

2.4 Information Retrieval Evaluation

The Annotation Subsystem supported by the work in this thesis currently supports annotations for search engine evaluation. The evaluation methodology supported by this system is based on the Cranfield Evaluation methodology [12], also called test-collection evaluation [13]. In such an approach, the main challenge is to create a test collection consisting of three parts: sample queries, sample documents, and relevance judgments. Collecting documents is relatively easy, but collecting queries and relevance judgments is challenging since it involves user effort. The traditional approaches to solving this problem rely on paying users to make annotations or running evaluation competitions as done in TREC [14]. The Annotation Subsystem of COLDS provides a more scalable way to solve this problem, which would enable creation of potentially many new data sets for evaluating search engines.

CHAPTER 3

COLDS AND ITS ANNOTATION SUBSYSTEM

This chapter provides brief introduction to COLDS and its annotation subsystem.

3.1 Overview

The basic idea of COLDS is to deploy software toolkits that contain algorithms for processing and analyzing big data sets on a cloud-computing infrastructure (currently Microsoft Azure). Currently COLDS integrated MeTA[1] toolkit to support big data analysis. The users can directly run experiments with algorithms and their parameters in the toolkit to learn the behaviors of algorithms on real data sets that are available on the cloud without downloading the data set locally.

3.2 Structure of COLDS Annotation Subsystem

The Annotation Subsystem of COLDS consists 3 different modules: search module, database module and annotation module. Search module directly connects with the toolkit, run the algorithms with user defined parameters on real data sets in the cloud, and return the results to the user. Database module record every action and stores them into the database. The stored data can be later used by other modules such as annotation module. Annotation module communicates search module and database module and provide the relevance judgement annotation functionality which can be further used to evaluate the algorithms and expand to leaderboard. The detail of search module and database module can be found in Liu[2] and Yu's[3] thesis.

3.3 Use of Annotation Subsystem

With the help of Annotation Subsystem, a leaderboard would be maintained by COLDS for every programming task to record the best the performance figures of the best performing algorithms submitted by learners or researchers. When the data set and task are new, such an infrastructure naturally supports research in data science as well, essentially removing the boundary of education and research. The search module makes it easy to run big data algorithms on real data sets. Moreover, as a research infrastructure, COLDS ensures reproducibility of all the experiments since they are all well documented in the infrastructure system and stored in the database module. Since a researcher does not have to re-produce any baseline methods (as they are already available via COLDS), the researcher only needs to implement and experiment with his/her new ideas, thus the workload of a researcher is minimized. In this sense, COLDS helps improve research productivity as well.

CHAPTER 4

DESIGN OF ANNOTATION MODULE

This chapter describes the design philosophy, challenges encountered and solutions to the challenges, and the structure of the annotation module.

4.1 Design Philosophy

The annotation module serves the key functionality of COLDS and all its modules. One functionality of the annotation module is to serve as a communication port between each different module. It will take the result from search engine module as input and save the annotation for each result as output to the database module. The annotation module also need to support the task assignment by interface module.

Since the module needs to satisfy those requirements to communicate each modules as well as serve its original functionality: do annotation on the query results, the module itself should be robust and scalable. In order to make the module itself more extensible, the design philosophy of the system also followed the principle of high cohesion and low coupling[15].

4.2 Challenges and Solutions

This section addresses the challenges encountered and solutions for those challenges

4.2.1 Design of Schema

The first challenge and decision choice for the Annotation Module, is the data schema used to store the annotation of each query and documents.

▼ (1) ObjectId("58f2a06e3c3ee16e0eac...	{ 6 fields }	Object
_id	ObjectId("58f2a06e3c3ee16e0eac0ab3")	ObjectId
annotator	ObjectId("58ede0a8fdfd400ddbf5da6b")	ObjectId
data_set	ObjectId("58f07b29dd83cd80dcb5936c")	ObjectId
query	ObjectId("58f175eb3c3ee15d45c67317")	ObjectId
doc	1	Int32
judgement	false	Boolean

Figure 4.1: The database schema for annotation module

The schema for the annotation module needs to be elegant, but at the same time stored all the information needed. The annotations have to store the information of the query, who the annotator is, what dataset the annotation is on and what relevance judgement the annotator made. At the same time, the schema needs to be efficient enough to support queries across multiple tables. As a result, the schema was designed to contain annotator, dataset, document ID and judgement, and made annotator and dataset as foreign key which followed the BCNF[16]. rule to satisfy all the requirement for the schema. Figure 4.1 shows an example of record stored in the database with such schema where '_id' is the object id provided by the database system, 'annotator' is the user who did such relevance annotation, 'data_set' is the experiment data set, 'query' is the query user ran on the data set, 'doc' is the document id inside the data set, and 'judgment' is the relevance judgement user made on this document.

4.2.2 Design of Data Structure

The second challenge and decision choice is the data structure used to store the annotation on the front end. The data structure should store each information needed in the schema. However, each annotation only has one annotator, data set and query information, but has multiple relevance judgement for each document inside the data set. The key point here is to efficiently store each information needed without redundancy. We need to store every relevance judgement for each document but at the same time store annotator, data set and query only once. To meet all the requirement, the data structure was designed in to a JSON object which stores annotator, data set and query as dictionary and relevance judgement as a list, each element contains the dictionary of document ID and relevance judgement of the document. In such a way, the data structure can both contain the information we needed

```
{
    "dataset": "58f07b29dd83cd80dcb5936c",
  "doc": [
    {"doc_id": 1,
       "judge" : "true"
     },
     {  "doc_id":2,
        "judge" : "false"
     },
     {  "doc_id":3,
        "judge" : "false"
     },
     {  "doc_id":4,
        "judge" : "true"
     }
     ],
  "query":  "58f175eb3c3ee15d45c67317",
  "user": "58ede0a8fdfd400ddbf5da6b"
  }
```


Figure 4.2: The data structure for annotation module

for annotation but as well as eliminate the redundancy.

4.3 Overall Structure

The overall structure of the annotation subsystem contains several parts to handle different requirements.

4.3.1 Interface

For educational purposes, we divided the web interface into two parts, the one for instructors and the one for students. The web interface for instructors can define annotation tasks for students. They can define the assignment on specific data set, the query content, and the algorithms and parameters they wanted to use. The user can choose 5 algorithms now. The default ranker is OkapiBM25[17].is the most commonly used ranker in search engine. It also supports Pivoted-length Normalization[18], Dirichlet Prior Smoothing[19], Jelinek-Mercer Smoothing[9]. and simply counting the number of appears of the keywords in the documents.

The web interface for students contains a list of tasks they ought to fin-

ish. Each task will do a pre-defined query on selected data sets with given ranker and parameters provided by the instructor. Students' interface can also support new queries defined by the students. The users can use this interface to start a new query and define the data set, query contents, algorithms and parameters by themselves, and do relevance judgement annotations on the returned results. The students' web interface can also serve as an experiments tool to discover queries on different algorithm with different parameters. Sample illustration of interface is showed in chapter 6.

4.3.2 Backend Service

The backend service serves as a middle layer to connect the other two parts. The interface sends its requests to the backend service. Backend service will parse the request and exact the information from the request. The backend service will then manipulate the data and reform the data structure to better fit the schema of the database and save the data to the database.

The request may also ask the data from database. The process is similar to the request to store data in database. The backend service will also parse the request to extract the information needed. Then, it will communicate with the database according to the requirements got from the request. Once got the required data in the database, the backend service will wrap the data into a JSON file and send it back to the interface with a status of 200.

4.3.3 Database

The database system serves will keep all the record into the data store. It will record the actions the user took and served as future use. When instructor assigns tasks, the database will store each task with the given data set, query, algorithm and parameter information. When the students start a task, the database will provide the task information stored by the instructor earlier. When the user does new queries, the database would also store the information which then can be further used for annotation purposes. The relevance judgment annotation is also stored in the database, and can be downloaded to run further experiment such as to compute the F-1 score and precision and recall of some ranker and parameters.

CHAPTER 5

IMPLEMENTATION DETAIL

This chapter introduces the implementation details of COLDS some code snippet.

5.1 Flask Server with RESTful API

The COLDS system was built with Python Flask framework. The server accepts each incoming HTTP request with RESTful API. With RESTful API, the server can handle each request in a clearer way, and each module can only handle the incoming HTTP request as API calls. The server can also use route render to separate each page. Python Flask is a light weighted micro web framework. It can support RESTful API and parse each request easily, and it is very easy to learn. As a result, users can easily modify the system according to their needs. Figure 5.1 shows a code snippet of Flask server.

5.2 Bootstrap with jQuery Front End

The front end of the annotation system was built with Bootstrap framework with jQuery library. The whole user interface webpage was made as a flask

```
api.add_resource(LoginAPI, '/login')
api.add_resource(AssignmentAPI, '/assign')
api.add_resource(AddQueryAPI, '/new_query')

@app.route("/instructor")
def instructor_page():
    return render_template("instructor.html")
```

Figure 5.1: Code snippet for Flask Server

```
function search(search_package){
    var data = {
        "ranker":search_package['ranker'],
        "query":search_package['query'],
        "num_results":search_package['num_results'],
        "params":search_package['params']
    };
    console.log(search_package['url']);
    $.ajax({
        type: "POST",
        dataType: 'json',
        url: search_package['url'],
        data: JSON.stringify(data),
        contentType: 'application/json; charset=utf-8'
    })
    .success(function(data){
        render_result(data);
    });
}
```

Figure 5.2: Ajex Reqeust for search

```
class Annotation(db.DynamicDocument):
    annotator = db.ReferenceField('User', required=True)
    data_set = db.ReferenceField('DataSet', required=True)
    query = db.ReferenceField('Query', required=True)
    doc = db.IntField(required=True)
    judgement = db.BooleanField(required=True)
```

Figure 5.3: MongoDB table as an Object

template, and each component was generated and rendered as a Bootstrap component. Once the webpage was rendered, user can input data in the web interface. When the user finished input data, the interface will send an ajex request with jQuery library. When the ajex requests returns, the callback function would be triggered and re-render the page with returned data packages. Figure 5.2 shows a code snippet of ajex request for search.

5.3 MongoDB and Mongoengine

COLDS used MongoDB as the data store. MongoDB is an open source, distributed NoSQL database. The database was hosted on Mongo Lab, which is a free online MongoDB. The backend service of COLDS used Mongoengine to connect and manipulate the MongoDB database. Mongoengine is a document object mapper from MongoDB to python. It let the user to manipulate the MongoDB in an ORM style, which instead of writing complex queries, user can operate MongoDB as the table is an object. Figure 5.3 shows a code snippet of mongoengine.

CHAPTER 6

APPLICATIONS OF THE COLDS' ANNOTATION SUBSYSTEM

This chapter describes some potential applications of COLDS and its Annotation Subsystem, and the benefits of using COLDS.

6.1 Education Application

COLDS and its Annotation Subsystem is naturally useful for big data education, especially Massive Online Open Courses (MOOCs). Using COLDS and its Annotation Subsystem, MOOC platform such as Cousera, Udacity and Edx can improve their existing data science courses by providing online programming assignment with COLDS and its Annotation Subsystem. Taking the online version of CS410 course of University of Illinois at Urbana Champaign as an example. Students can register at COLDS platform to do online programming assignment. Figure 6.1 shows a sample illustration of signup page.

The TA and instructors of the course can upload real data sets to the platform and assign tasks to each student. The students of the course can

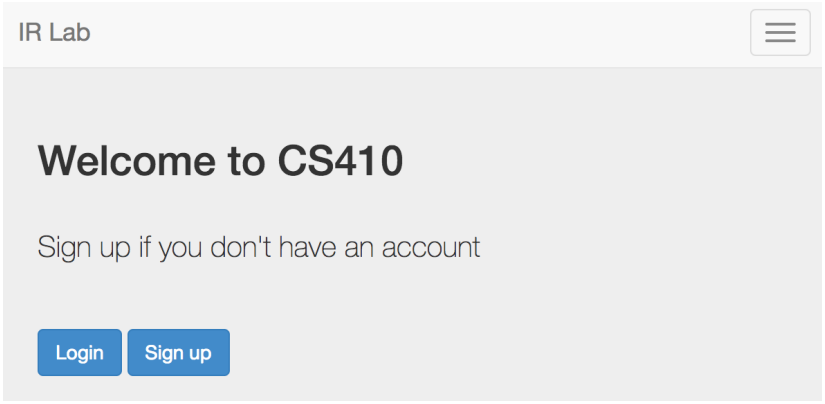

Figure 6.1: Signup Page

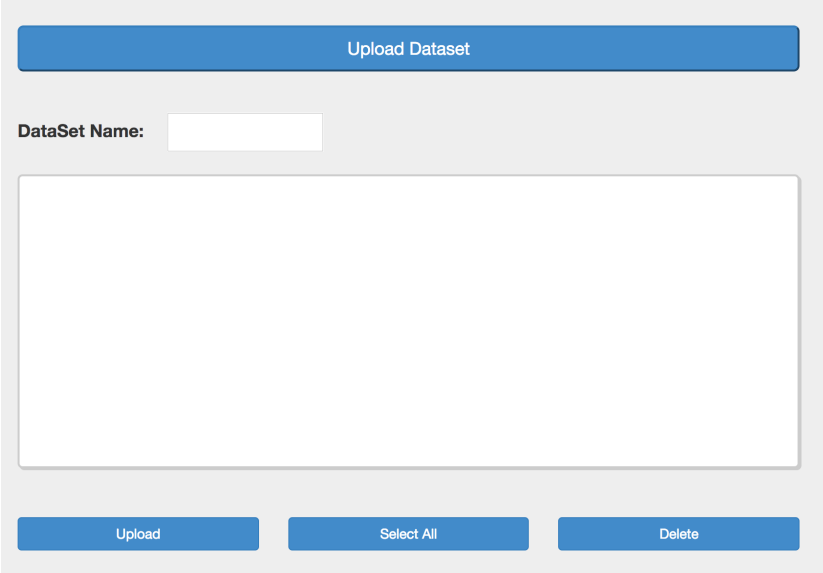

Figure 6.2: Upload Page

select assignment on their side and finish them. The students can also try to run the algorithms on these data sets to do some experiments and learn from them with the platform. Figure 6.2 shows a sample illustration of upload page. Figure 6.3 shows a sample illustration of students' page. Figure 6.4 shows a sample of search results page.

6.2 Benefits of COLDS' Annotation Subsystem

From education perspective, besides enabling big data programming assignments to be done at large scale, COLDS ensures the students learn skills that would be directly useful for solving an industry problem, thus minimizing the distance between education and applications in big data. From an industry partners perspective, contributing data sets to COLDS has multiple benefits, including visibility, training highly relevant labor force for the company, access to potential candidates for hiring, and annotating their data sets via crowdsourcing annotations using annotation assignments. This last benefit would enable new research to be done in data science also, but requires a general annotation system to be included in COLDS.

Figure 6.3: Student Page

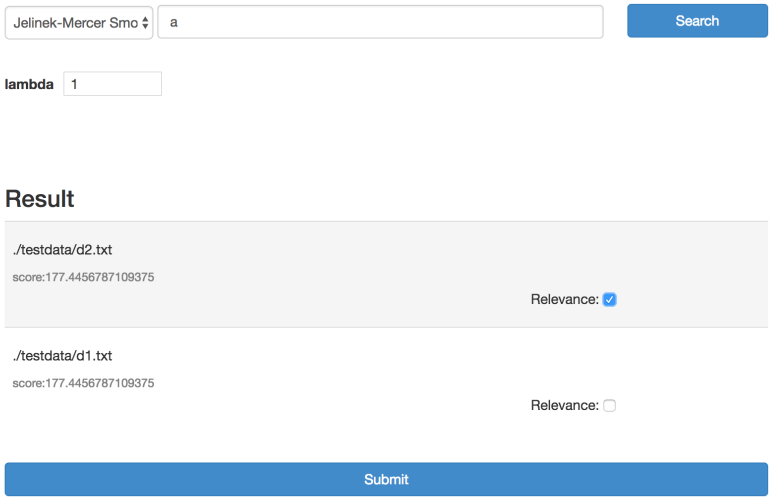

Figure 6.4: Search Page

CHAPTER 7

SUMMARY

With the trend of big data, the industry required more data scientists and engineers with such skills more than ever. Big data education has gained more popularity than before. However, present online education platforms such as Cousera, Udacity and Edx lack the ability to provide online programming assignments. This thesis presented COLDS and its Annotation Subsystem, which is a novel cloud based general infrastructure to online assignment efficiently at a low cost. This thesis mainly focused on design and implementation of the annotation module of COLDS and showed some example application with sample illustrations.

In conclusion, COLDS and its annotation subsystem has a great potential in future big data online education and can make big impact.

REFERENCES

[1] S. Massung, C. Geigle, and C. Zhai, "Meta: A unified toolkit for text retrieval and analysis," *ACL 2016*, p. 91, 2016.

[2] C. Liu, "Design and implementation of the database module in colds for data annotation," M.S. thesis, University of Illinois at Urbana Champaign, Urbana, 2017.

[3] X. Yu, "Design and implementation of the search module in colds," M.S. thesis, University of Illinois at Urbana Champaign, Urbana, 2017.

[4] Tom Kalil and Fen Zhao, "Unleashing the power of big data," 2013. [Online]. Available: https://obamawhitehouse.archives.gov/blog/2013/04/18/unleashing-power-big-data

[5] "Ml comp," 2013. [Online]. Available: http://mlcomp.org/

[6] "Kaggle," 2017. [Online]. Available: https://www.kaggle.com

[7] H. Fang, H. Wu, P. Yang, and C. Zhai, "Virlab: A web-based virtual lab for learning and studying information retrieval models," in *Proceedings of the 37th International ACM SIGIR Conference on Research & 38; Development in Information Retrieval*, ser. SIGIR '14. New York, NY, USA: ACM, 2014. [Online]. Available: http://doi.acm.org/10.1145/2600428.2611178 pp. 1249–1250.

[8] L. Azzopardi, Y. Moshfeghi, M. Halvey, R. S. Alkhawaldeh, K. Balog, E. Di Buccio, D. Ceccarelli, J. M. Fernández-Luna, C. Hull, J. Mannix, and S. Palchowdhury, "Lucene4ir: Developing information retrieval evaluation resources using lucene," *SIGIR Forum*, vol. 50, no. 2, pp. 58–75, Feb. 2017. [Online]. Available: http://doi.acm.org/10.1145/3053408.3053421

[9] J. Howe, "The rise of crowdsourcing," *Wired magazine*, vol. 14, no. 6, pp. 1–4, 2006.

[10] H. Su, J. Deng, and L. Fei-Fei, "Crowdsourcing annotations for visual object detection," in *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, vol. 1, no. 2, 2012.

[11] A. Wang, C. D. V. Hoang, and M.-Y. Kan, "Perspectives on crowdsourcing annotations for natural language processing," *Language resources and evaluation*, vol. 47, no. 1, pp. 9–31, 2013.

[12] K. S. Jones, *Readings in information retrieval*. Morgan Kaufmann, 1997.

[13] M. Sanderson et al., "Test collection based evaluation of information retrieval systems," *Foundations and Trends® in Information Retrieval*, vol. 4, no. 4, pp. 247–375, 2010.

[14] E. M. Voorhees, D. K. Harman et al., *TREC: Experiment and evaluation in information retrieval*. MIT press Cambridge, 2005, vol. 1.

[15] S. McConnell, *Code complete*. Pearson Education, 2004.

[16] E. F. Codd, *Recent Investigations in Relational Data Base Systems*. IBM Thomas J. Watson Research Division, 1974.

[17] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford et al., "Okapi at trec-3," *Nist Special Publication Sp*, vol. 109, p. 109, 1995.

[18] A. Singhal, C. Buckley, and M. Mitra, "Pivoted document length normalization," in *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1996, pp. 21–29.

[19] C. Zhai and J. Lafferty, "A study of smoothing methods for language models applied to ad hoc information retrieval," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2001, pp. 334–342.