Yiwen Jiang. Implementation of a High Recall Interactive Literature Retrieval System. A Master's Paper for the M.S. in IS degree. April, 2019. 43 pages. Advisor: Yue Wang

This project-based study implements a high recall and high precision interactive literature retrieval system based on the ReQuery-ReClassify (ReQ-ReC) framework proposed by Wang et al. in 2014. The study summarizes the challenges and difficulties of current methods of literature retrieval and review in achieving high recall in addition to high precision. Following the double-loop mechanism of the ReQ-ReC framework, the project applies the methodology of system design, database design and user interface design to turn the framework into a real-world web application. Heuristic evaluation for the user interface design indicates that the system is user-friendly and can be integrated with literature retrieval systems like PubMed.

Headings:

High-recall search

Literature retrieval

Human-in-the-loop machine learning

System design

Full-stack web development

IMPLEMENTATION OF A HIGH RECALL INTERACTIVE LITERATURE RETRIEVAL SYSTEM

by
Yiwen Jiang

A Master's paper submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina

April 2019

Approved by

____

Yue Wang

Table of Contents

1 Introduction

1.1 Motivation

Information retrieval is ubiquitous in our lives. We search for articles through online literature retrieval systems to view recent studies, we search for interested posts in Twitter, and we search for suggested routes when we have a trip, etc. People perform different tasks on retrieval systems and gather information from the search results. The precision and recall of the system will influence the user experience and the efficiency and effectiveness of completing the tasks, where precision and recall are two measures of relevance. Precision stands for the fraction of relevant instances among the retrieved instances while recall stands for the fraction of relevant instances that have been retrieved over the total amount of relevant instances[1]. In some scenarios, we need high precision, while in others we require high recall. Commonly, we are facing more precision-oriented scenarios, like most search engines typically return a limited number of results that are the most relevant to the user's typed query based on some ranking functions, which satisfies high precision. However, scenarios also exist in which the searcher requires both high precision and high recall. Such scenarios are not uncommon in real life, exemplified by social searches, medical searches, legal searches, market research, and literature review searches.

To address this issue, one recent research study introduced a ReQuery-ReClassify framework[2] which aims to achieve both high precision and high recall. The basic idea of the framework is to distribute the burden of maximizing both the precision and recall to a set of queries and a classifier, where the queries are responsible for increasing the recall of relevant documents retrieved and the classifier is responsible for maximizing the precision of documents retrieved collectively by all of the queries in the set. The framework features a double-loop mechanism: the inner-loop classifies the retrieved documents, actively collects user feedback, and improves the classifier (ReClassify); the outer-loop generates new queries (ReQuery) and iteratively adds newly retrieved documents into the work set. The research conducted empirical experiments to evaluate the effectiveness of the framework and its instantiations. Their experiments show that some instantiations would achieve a 20%-30% improvement of mean average precision and R-precision on most data sets, with the largest improvement up to 150% over classical iterative relevance feedback. The proposed framework would be a solution to those retrieval scenarios which require both high precision and high recall.

Inspired by this research, this project-based study aims to build an interactive retrieval and learning system which would implement a "human-in-the-loop" interactive text search and classification system based on the ReQuery-ReClassify framework mentioned above. "Human-in-the-loop" here refers to an adaptive system that incorporates user feedback.

1.2 Objective

This project aims at developing a high precision and high recall literature retrieval system which serves the following key functions:

- Retrieve relevant documents for user based on their typed queries.

- Allow the user to explicitly label search results based on their own understanding and judgments.
- Get user labels and use them to build the classifier and reclassify retrieved documents.
- Give user suggested query terms based on relevance judgments.
- Let user view/edit the suggested query and compose new queries.

The system consists of six key components: user interface, search engine, data storage, document classifier, document selector and query generator. The users interact with the system through a web-based user interface. The search engine gathers user's queries and returns search results. The data storage stores data transferred in the system and support other components. The document classifier learns from users' relevance feedback on search results and improves precision. The document selector selects which document to let user label on. And the query generator constructs new queries in order to improve recall. As shown in Figure 1, the process of the system follows the double-loop mechanism mentioned above.

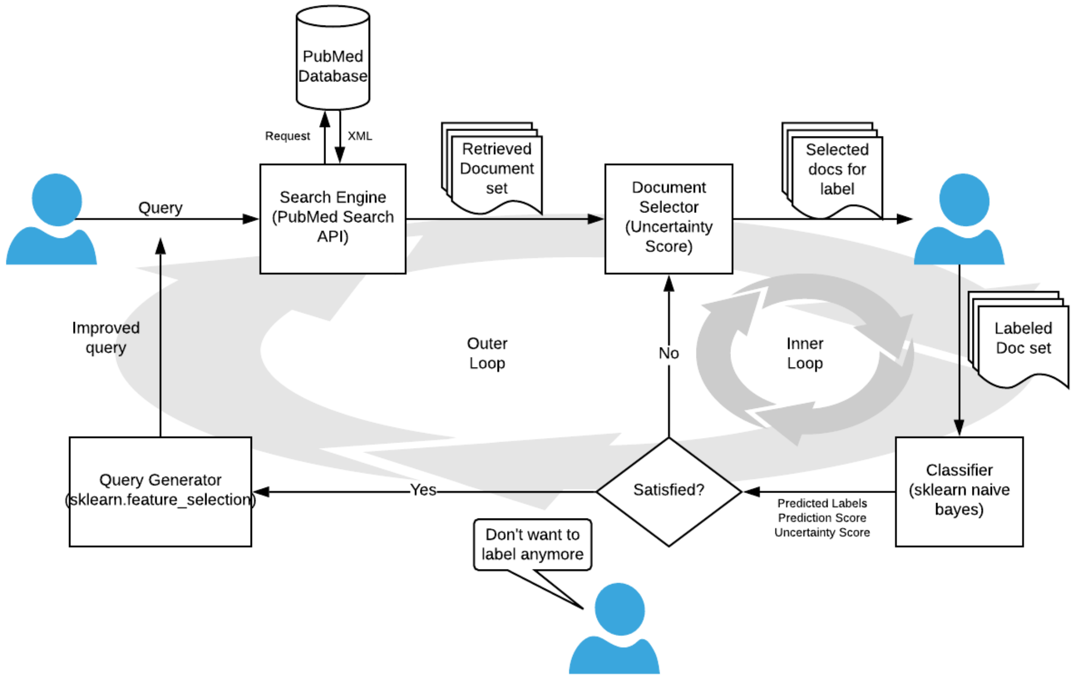


Figure 1 System Workflow

The development of the system follows the pattern of software development life cycle (SDLC). The produced high precision and high recall retrieval system can be integrated with online search engines to improve their search results and save user's time and efforts. Specifically, we consider integrating with biomedical literature retrieval systems such as PubMed, which are used by health science librarians to perform systematic literature review.

This study aims to answer the following research questions:

RQ1: Did previous literature retrieval methods/systems bring both high precision and high recall results and are easy to apply to real world applications?

RQ2: Does the built-up system implement the ReQ-ReC framework successfully?

RQ3: Is the system practicable enough to be embedded into real retrieval systems like PubMed?

The following chapters will first look at previous studies on methods of systematic review, will then introduce the system design and user interface design of the implementation of the high recall and high precision interactive literature retrieval system, and will next evaluate the user interface design and make conclusions accompanied by limitations at the end.

NOTES

1 Precision and recall, Wikipedia, https://en.wikipedia.org/wiki/Precision_and_recall

2 Li, C., Wang, Y., Resnick, P., & Mei, Q. (2014, July). Req-rec: High recall retrieval with query pooling and interactive classification. In Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval (pp. 163-172). ACM.

2 Prior Work

This chapter introduces and summarizes prior researches working on “human-in-the-loop” mechanisms implemented in information retrieval, previous workflow of systematic review, and studies on technology-assisted review.

2.1 “Human-in-the-loop” mechanisms

Recent researches have been paying much attention to “interactive systems” and “human-in-the-loop” in all kinds of retrieval systems, including literature retrieval system, image retrieval system, etc. The concept “human-in-the-loop” leverages both human and machine intelligence to create machine learning models [1]. In this mechanism, humans are directly involved in training, tuning and testing data for a specific machine learning algorithm. Such mechanism would let the machine learning model behind the system keep improving continuously and provide better results through the whole process. Applications which involve human-in-the-loop mechanism necessitate greater transparency in machine learning models for experts to understand and trust their decisions [2].

Relevance feedback-based approaches are commonly used methods in such mechanism. Relevance feedback is an automatic process, introduced over 20 years ago, designed to produce improved query formulations following an initial retrieval operation [3]. Several studies proposed relevance feedback architectures and frameworks in image retrieval, where human and computer can interact with each other to improve the retrieval performance [4,5,6]. We can observe that relevance feedback, human in the loop mechanism have already successfully been applied in image retrieval systems [7], while there still exists limitation on their application in literature retrieval system, which also has high demand on reaching high retrieval performance.

2.2 HRR (High Recall Retrieval) problem

Systematic/literature review plays an important role in any academic research, which provides an overview of what's been studied and written about a specific topic. From the perspective of librarians working on reviews, they are aiming at finding the full set of relevant documents(achieve high recall in addition to high precision) in order to be as comprehensive as possible to cover all the previous work, find out state-of-the-art evidence to guide their further work directions, which is definitely a hard and time-consuming task[8,9,10,11]. The existing HRR methods have been far from satisfactory to make them enumerate all relevant documents, which is because not only the sheer volume of documents inevitably including noises (non-relevant documents) but also the threshold measurements have been inadequately adopted [8]. Prior researches proposed several methods and models in order to solve such problems. [9] demonstrated how to optimize performance at high recall levels systematic review in public health field when using linear SVMs for ranking. Specific techniques included feature engineering that exploits facets used in the human querying process; iterative retraining of models using sampled annotations, and processing documents with missing fields using separately trained classifiers, etc. [13] also mentioned the demand to apply query expansion to enhance further the search strategy and pointed.

Prior works proposed many strategies on increasing precision or recall. However, traditional systematic reviews find it hard to balance between precision and recall. We can observe that due to the HRR problem, current systematic review workflows (Figure 2, 3) are complex and time-consuming to some extent [11, 12]. Researchers need to modify their queries for many times in order to reach the high recall goal, and sometimes the query would be very long and redundant. Solutions which combine strategies to both increase high precision and high recall still need to be explored.

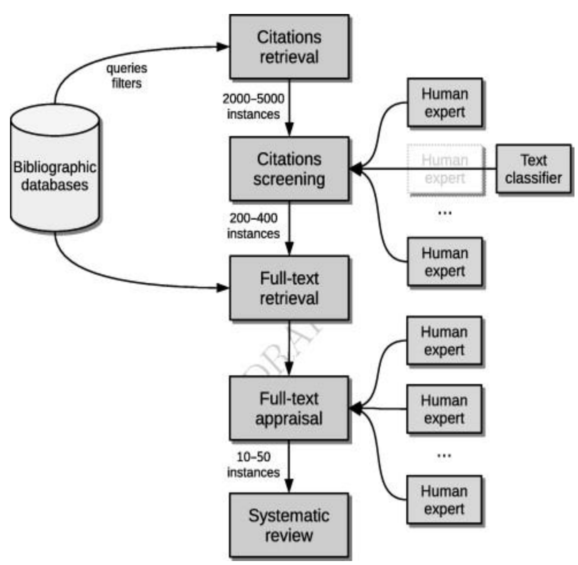


Figure 2
Overview of the traditional process to produce a systematic review modified by the inclusion of automatic text classification to the citation screening phase from [11].

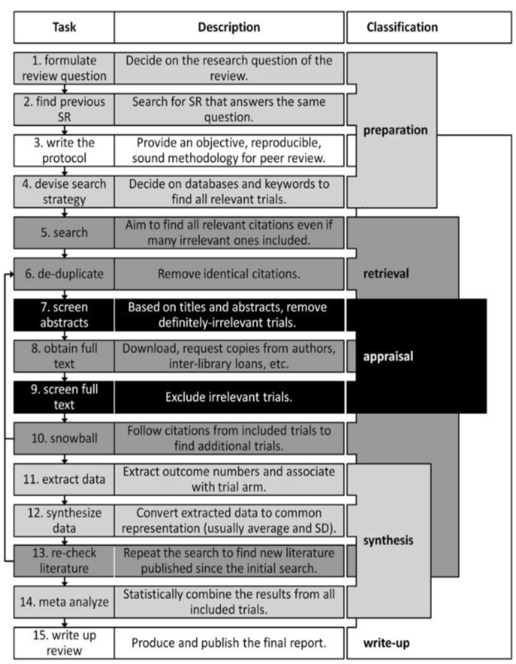


Figure 3
Existing methods for systematic reviews follow these steps with some variations from [12]. Not all systematic reviews follow all steps. This process typically takes between 12 and 24 months.

2.3 Technology-assisted review

With the help of internet and technology, online IR portals have been useful tools for researchers to retrieve information and literatures. Currently, common online IR systems like Google Scholar does not provide necessary elements for systematic scientific literature retrieval such as tools for incremental query optimization, export of many references, a visual search builder or a history function [14]. [13] also pointed out that an automatic query expansion based on the users' interests is a desirable feature of search engine, but most search engines do not support this feature beyond mapping selective query terms to ontology or thesaural headings (e.g., PubMed).

In conclusion, "human-in-the-loop" mechanism could be used to help address systematic/literature review with HRR problem. We could use technology to assist review to facilitate manual works. A user-friendly literature retrieval system which could reach both high precision and high recall using relevance feedback and query expansion is needed.

3 System Design of High Recall Interactive Literature Retrieval System

This chapter introduces modular design, database design, and use case design of the high recall and high precision literature retrieval system. In order to better understand the design, some explanations on concepts appear in this chapter, assumptions and technology stack used in the system are needed:

Concepts and Definitions

- Inner-loop: The inner-loop refers to one part of a complete search process. It starts from type query, view results, then label results, train classifier, and end at get prediction scores from classifier. Inner-loop will reclassify and re-rank search results based on prediction scores given by the classifier in order to get higher precision.
- Outer-loop: The outer-loop refers to the other part of a complete search process. It uses suggested query terms returned by the feature selection function and then collects more documents in order to increase recall.
- Task: A task refers a complete search process in the system. In another word, a task consists of several iterations of inner-loop and outer-loop (see Figure 13 Activity Diagram for details) to achieve the goal of getting high precision and recall.

Assumptions

- In order to reduce complexity, assume that there's only one user in one search task;

- User uses the system to achieve high precision and high recall retrieval;
- User will not be willing to view and label more than 1000 articles;
- User could determine whether a document is related or not based on only the abstract of the document.

Technology Stack

- Client Side: JavaScript, jQuery, HTML, CSS
- Server Side: PHP, MySQL Database, Python (document classifier) scikit learn library

3.1 Modular Design

Based on the process of the framework, I used modular design to subdivide my system into five modules: search engine module, data storage module, document classifier module, document selector module, and query generator module. The following subsections will introduce each module's responsibility to the whole system and briefly explain how those modules are implemented by technical skills/framework.

3.1.1 Search Engine Module

The search engine module is designed to return a set of documents from the full document set based on user input query. The Entrez Programming Utilities (E-utilities) provided by NCBI (National Center for Biotechnology Information) are the public API to the NCBI Entrez system. Developers can use the API to access Entrez databases including PubMed, PMC, etc. In this system, I chose the PubMed database as the full document set.

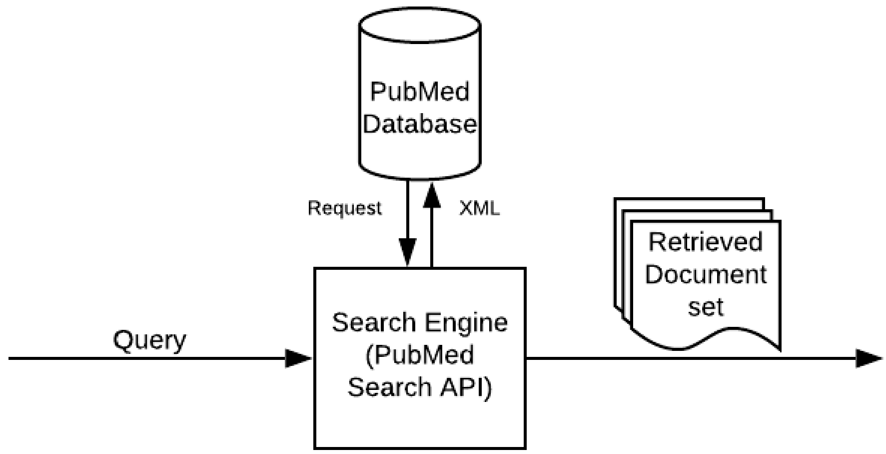


Figure 4 Search Engine Module

To implement search engine module, I used Ajax method, which could change content dynamically without the need to reload the entire page. In the front-end interface, when user click on the search button, a XMLHttpRequest object will be created by JavaScript. The XMLHttpRequest object will then send a request to the PubMed server. The PubMed server will process the request and will send a response back which contains a list of document data including document id (PMID), title and abstract. The response will be processed by JavaScript and then displayed on the result page.

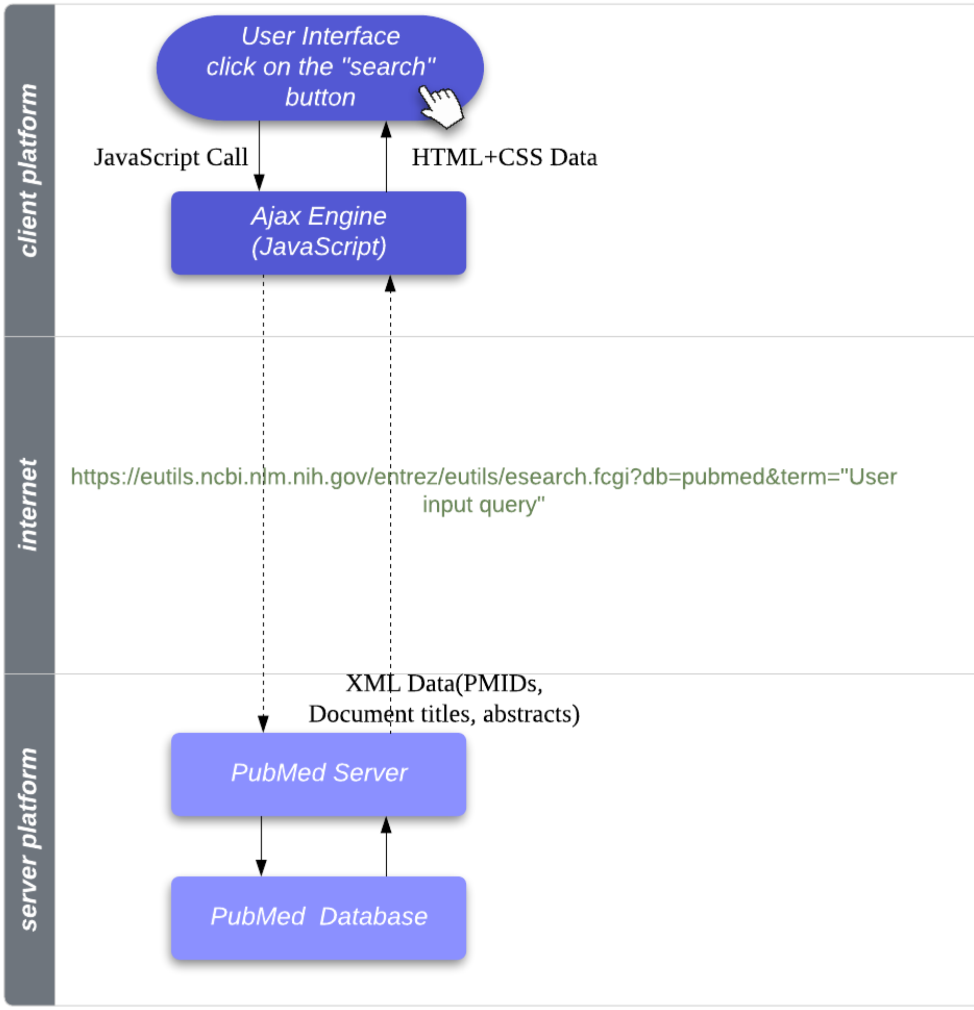


Figure 5 Using Ajax to update web page with document data dynamically

3.1.2 Data Storage Module

The data storage module is designed to store data needed in the workflow (see Figure 13 in section 3.3.2) in order to support the operation of the system. There are four cases which need the support of data storage module:

- Start inner-loop: insert 1000 document data, update query
- Update label: update user labels
- Train data: update prediction results

- Outer-loop: insert new retrieved documents into table exclude duplicates

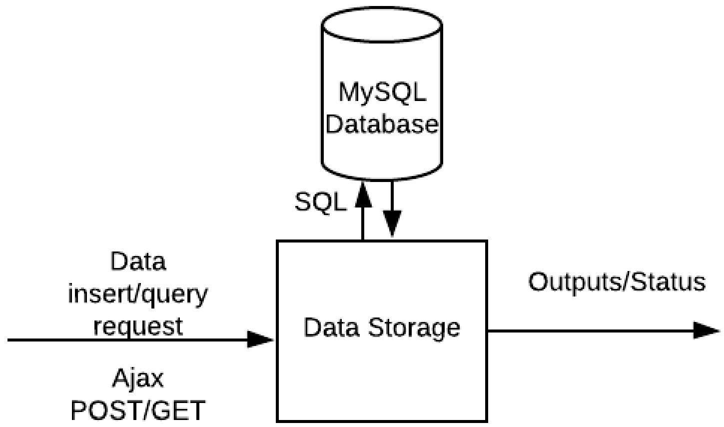


Figure 6 Data Storage Module

The data storage module is implemented by PHP and MySQL on the server side. When the front end needs to insert data or query data, it will send an Ajax request to the PHP script on the server side. The PHP script will connect to the database and execute data insert or query. Database design will be introduced in section 3.2.

3.1.3 Document Classifier Module

The document classifier module is designed to re-classify all the retrieved documents into relevant or non-relevant category based on user labels in order to increase the precision. The classifier would learn from the labeled document set and train itself.

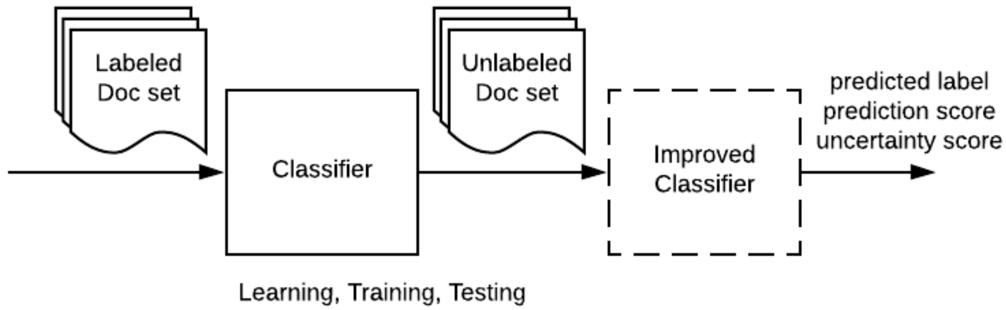


Figure 7 Document Classifier Module

The system currently uses the Naive Bayes model as the document classifier. The classifier is implemented by Python function using scikit-learn library [17]. The Python function will first preprocess the labeled documents, transform them into TF-IDF vectors and then build the classifier. When the re-classification completes, it will output predicted label, prediction score and uncertainty score for each unlabeled document.

Outputs	Explanation
Predicted label	The predicted category of each document Values: relevant or non-relevant
Prediction score	The posterior probability of "relevant" category of each document Value: Score(prediction) = P(relevant|doc)
Uncertainty score	The uncertainty of the classifier for a specific classification. Value: Score(uncertainty) = 1 - max{P(relevant|doc), P(non-relevant|doc)}

Table 1: Outputs of Document Classifier and Explanations

3.1.4 Document Selector Module

The document selector module is designed to select documents from retrieved document set that are yet unlabeled to let user to label based on their own judgement. For each document retrieved by the search engine module, uncertainty score would be calculated after one iteration of inner-loop. The document selector, which aims to maximize the learning rate of the classifier, should return the most uncertain documents for user to label in every iteration of the inner-loop. At the beginning of each search task, since there are no judged documents, the document selector could return the top documents ranked by the retrieval function, which are ranked by document IDs.

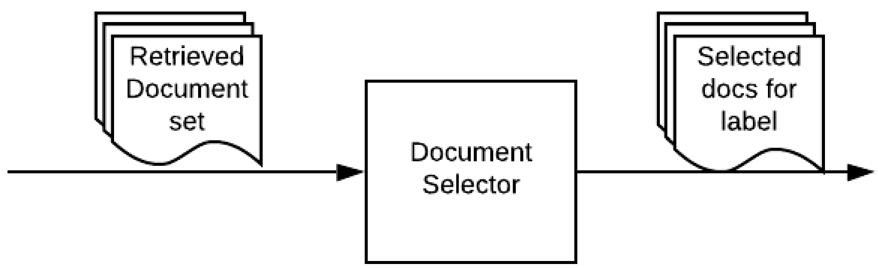


Figure 8 Document Selector Module

3.1.5 Query Generator Module

The query generator module is designed to expand the query in order to increase the recall in the outer-loop. It will generate 20 best features which are correlated with "relevant" category and are most useful to the classification based on labeled document set. User may consider using these most useful features to make up a new query in the next iteration of the loop to retrieve more related documents and increase recall.

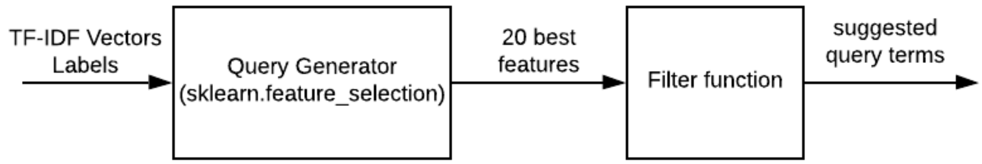


Figure 9 Query Generator Module

The query generator module is implemented by feature_selection module of scikit learn library. The feature_selection module provides the SelectKBest class which can be used with a suite of statistical tests to select a specific number of best features. Mutual Information is a common statistical test method usually used in classification tasks. A feature with higher mutual information in one target class means that the feature makes more contribution to the classifier in making the correct classification decision on that target class and is more useful in that class. Thus, I chose mutual information as the statistical test to select 20 best features which are most useful to the classification. Then, I used a filter function to filter out those features which are correlated with the "non-relevant" class since we only want features correlated with "relevant" class to be considered as suggested query to user.

3.2 Database Design

3.2.1 Data Entities

Data transferring in system are stored in MySQL database. There are four types of data entities in the workflow of the system which needed to store in order to support the operation of the system:

- Search Task: A search task entity stands for a finished searching task performed by a user. It records all the queries executed during one search task in order to raise the recall.
- Query_Document: A query_document entity stands for a retrieved document returned from the search engine based on a specific query.
- User_Document: A user_document entity stands for user's label for a document returned from the document selector in the inner-loop.
- Document_Classifier: A document_classifier entity stands for a set of attributes of a document returned from the document classifier. In each iteration of inner-loop, the classifier would learn from the user label and reclassify all the retrieved documents. Returned attributes include the predicted label of the document, the prediction score and uncertainty score of the prediction.

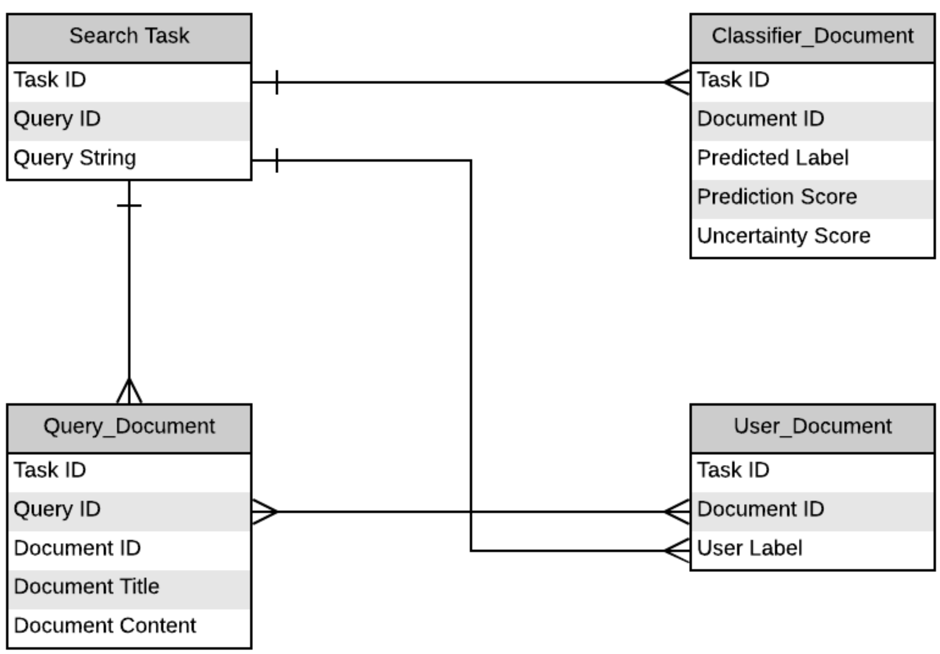


Figure 10 Entity Relationship Diagram

3.2.2 Database Tables

Those four types of entities are mapped into two database tables: queries and articles.

- Queries: The "queries" table records all the queries executed during search tasks in order to raise the recall. One could retrieve all queries within one search task using task ID. One could also retrieve a specific query in one outer-loop of a task by using query ID and task ID.
- Articles: The "articles" table stores all search results of multiple search tasks. It stores all attributes of an article needed by the system, including article title, article abstract, user label, predicted label, prediction score and uncertainty score. An article could be uniquely identified by task ID and article ID.

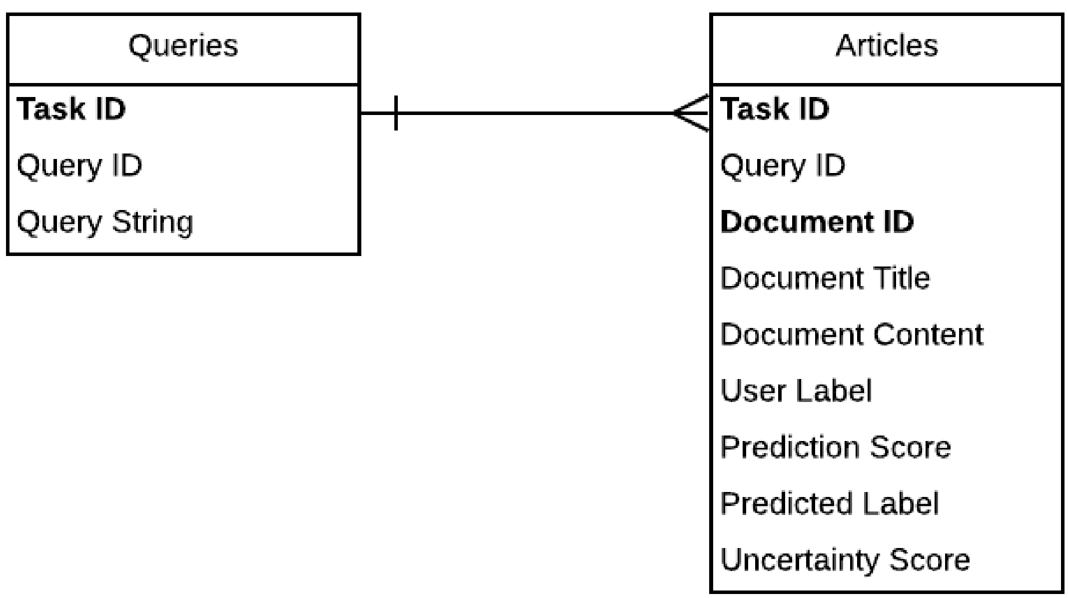


Figure 11 Database Schema Diagram

Columns (Field Name)	Explanation
Task ID (taskid)	Identifier of a search task
Query ID (queryid)	Identifier of a query
Query String (querystr)	The string of a query
Primary Key: Task ID	E.g. (0,0, "cancer")

Table 2: Fields in table "Queries"

Columns	Explanation
Task ID (taskid)	Identifier of a search task
Query ID (queryid)	Identifier of a query
Document ID (artid)	The document id of a retrieved document returned by PubMed search API.
Document Title (title)	The title of a retrieved document.
Document Content (abstract)	The abstract of a retrieved document.
User Label (label)	The label of a retrieved document labeled by user. Using numbers to represent the label. 1 refers to "Yes", means the user thought this article is relevant, while 3 refers to "No" means the user thought this article is non-relevant. 0 means the user did not label this document.
Prediction Score (score)	The posterior probability of "related" category of a retrieved document returned by the document classifier.
Predicted Label (pred_label)	The predicted label of a retrieved document returned by the document classifier.
Uncertainty Score (uncert_score)	The uncertainty of the classification result returned by the document classifier.
Primary Key: (taskid, artid)	E.g. (2, 0, 340828, "How I do it--plastic surgery: practical suggestions on facial plastic surgery. The use of upper eyelid skin grafts in the head and neck.","Recognition of the allergic individual…", 0, 0.585784, Related, 0.414216)

Table 3: Fields in table "Articles"

3.3 Use Case Design

This section will introduce how the user is expected to interact with the system and user's workflow within the system.

3.3.1 Use Case

User is the operator of the system, he/she needs to perform several cases in order to finish the search task and get high precision and high recall. The use case design of the system is shown in Figure 12.

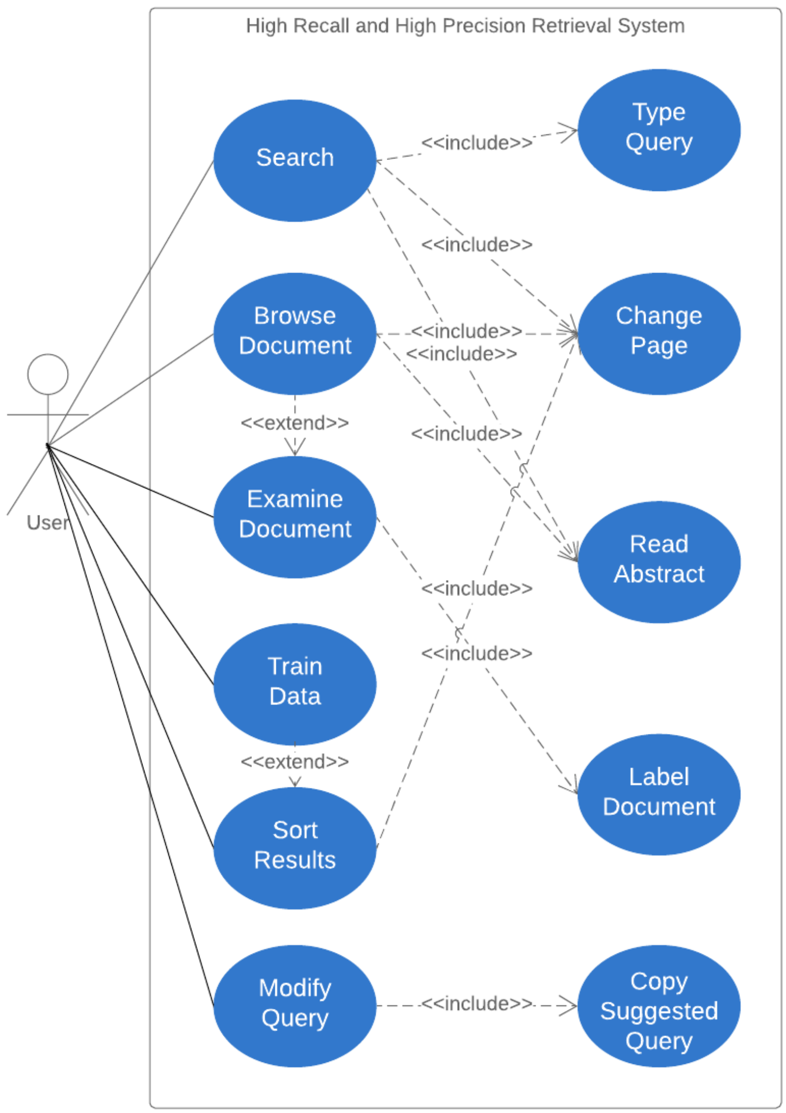


Figure 12 Use Case Diagram

Below table gives detailed explanation of activities behind each use case.

Case	Activities
Search	Type query into input box and then get search results from PubMed database.
Browse Documents	Browse search results, view document titles and abstracts.
Examine Documents	Determine whether a document is relevant or non-relevant and then label documents.
Train Data	Send labels to the document classifier and then reclassify all the results.
Sort Results	Get predicted results from the document classifier and sort results by prediction score, uncertainty score or other fields.
Modify Query	Get suggested query terms from the document classifier, modify the query and search again.

Table 4: Activities behind each use case

3.3.2 User Workflow

In order to run the system, user (front-end user interface), controller (back-end functions) and database need to work together. These three components need to transfer parameters and data to each other to support each use case. Figure 13 shows the activity diagram of the system.

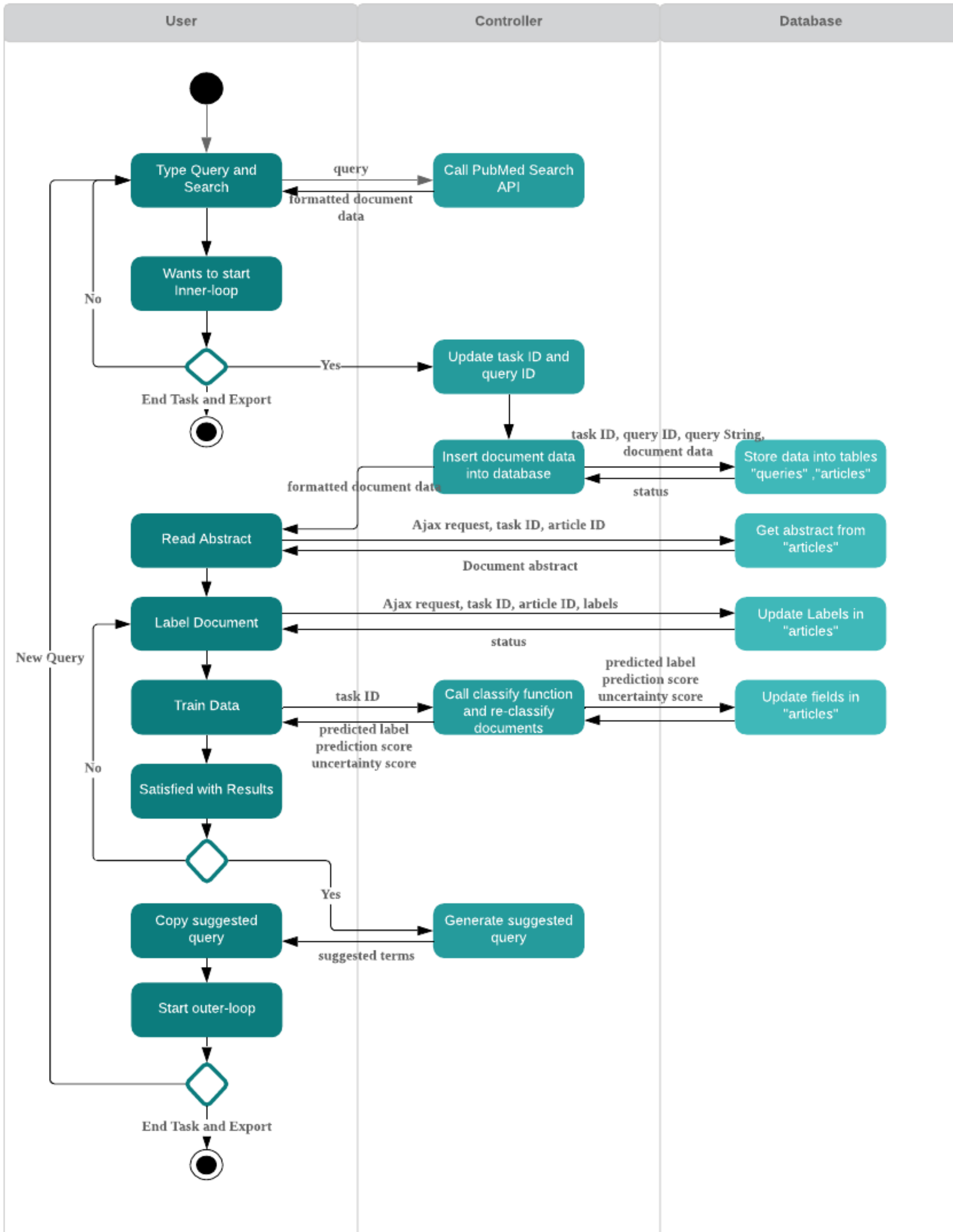


Figure 13 Activity Diagram

The workflow starts from a user types query into the system and triggers search event. The PubMed search API will respond a list of document data based on the query. In this session, no data will be inserted into the database. All the results shown to user will be extracted from the response from PubMed search API. When user wants to start one iteration of the inner-loop, first 1000 (or less than 1000, depending on the total number of results) document data would be inserted into the database. All the data needed in the rest of the task would be extracted from the database. User then could label the documents and upload the labels into the database. After submitting labels into the database, user could trigger classifier to reclassify all the documents. Once the reclassification finished, the classifier will update prediction score, predicted label and uncertainty score fields in the database and display to user. User could also sort results based on those fields and label documents with high uncertainty score to maximize the learning rate of the classifier. When user is satisfied of the precision or does not want to label anymore, she/he could stop labeling and training. The classifier will also generate suggested query terms based on feature selection. User could copy suggested query terms and paste to the input box to start one iteration of outer-loop to increase recall. User could trigger end task to stop the search task and export results.

4 User Interface Design

Based on the use case design section in 3.3, the user interface is divided into six widgets as shown in the below Figure 14.

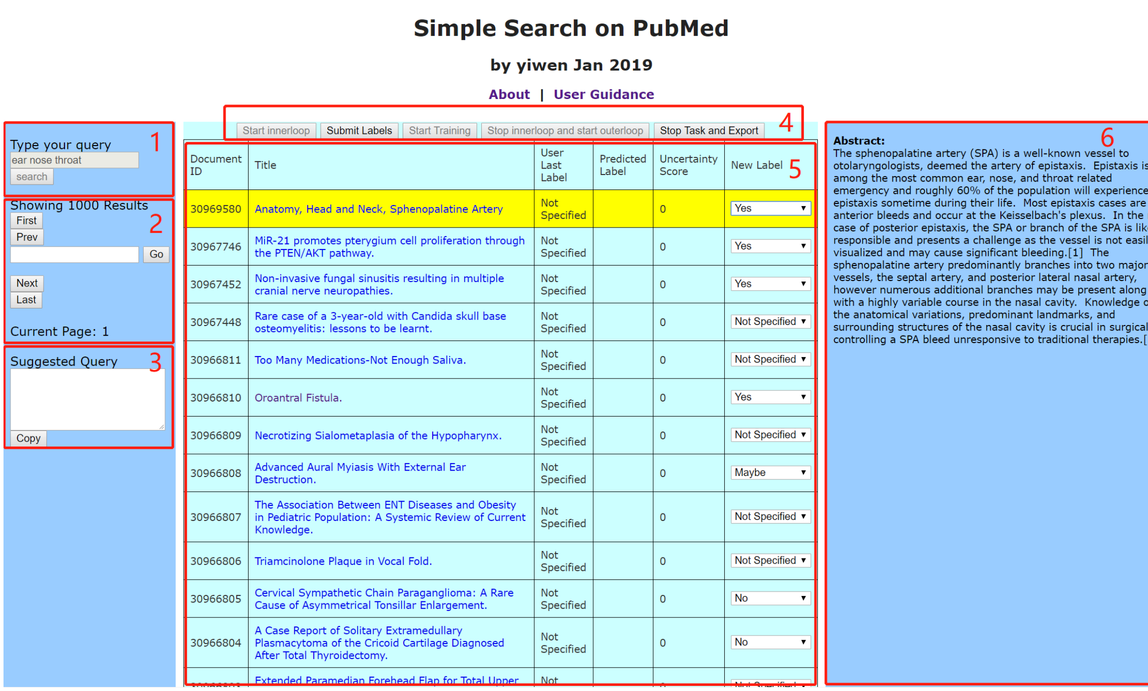


Figure 14 User Interface of the system

1: Query Form

Query Form is linked to the search engine module, which consists of a input box to let user type in queries and a search button which triggers PubMed search API, get response and extracts data from the response.

2: Page Control

Since there are many search results to display, the results need to be paged. The Page Control is designed to let user view results in different pages. Four buttons are given: “First”, “Last”, “Next”, “Prev” which let user jump into the first, last, next or previous page.

3: Query Suggestion

Query Suggestion consists of a text area and a copy button. It will gather suggested query terms from the query generator module and display in the text area. When user click on the copy button, it will automatically copy the query terms into user’s clipboard so that user can paste them into the input box in Query Form.

4: Operation Menu

Operation Menu provides five buttons to user in order to proceed the search process: start innerloop, submit labels, start training, stop innerloop and start outerloop, stop and export. Table 5 shows detailed explanation about functions of each button.

Button	Triggered Events
Start innerloop	Insert first 1000 retrieved document data into table “articles” to prepare for the inner-loop. Insert current query into “queries” table. New task ID and query ID will be created and maintained until user triggers stop task.
Submit Labels	Collect labels from user’s selection of each dropdown menu in 5 and update into the table “articles”.
Start Training	Use user labels to train the classifier and then re-classify all selected documents. When the re-classify completes, the classify results will be displayed on 5 and suggested query terms will be displayed on 3.
Stop innerloop and start outerloop	Indicate that the user does not want to label any more currently and wants to use suggested query in 3 to start outer-loop.
Stop Task and Export	Indicate that the user wants to stop current search task. Search results will be automatically exported as csv format and download to user.

Table 5: Buttons and Triggered Events

5: Result Panel

Result Panel displays retrieval results in a table format. Columns in the table include document id (the id in PubMed database), document title and other attributes of the documents needed by current sub-task (prediction score, uncertainty score, predicted label, etc.). The document title contains a link which will redirect user to the PubMed page of the document to get more information. The last column contains a list of drop-down menus which let user to label the results. When user hovers mouse on the row of a specific document, the row would be highlighted and be changed back to original when user moves out the mouse. The document classifier will update the prediction score, uncertainty score, predicted label of each result after reclassification. By clicking on the heading of the table, user can sort the results by the selected fields.

6: Text Panel

Text Panel shows the abstract of a specific document when user hovers mouse on a row of results displayed in the Result Panel. User would label the document as related or non-related after viewing the abstract of the document.

5 Heuristic Evaluation for User Interface Design

The user interface design could be evaluated by 10 usability heuristics raised by Jakob Nielsen in 1994. This chapter analyses how the user interface design of the system meets the 10 heuristics and thus proofs the usability of the system.

- **Visibility of system status**

The system will block the UI during the process of interacting with the database, including inserting data and training data to avoid inappropriate actions from user which would influence the process of transferring data. After each sub-task finished, the pop-up alert window will tell user about the status of the task. Thus, it would always keep user informed about what is going on and would provide appropriate feedbacks when needed.

- **Match between system and the real world**

The system uses understandable and simple language which make it easier for user to use if the user understands the double-loop mechanism the system follows. The order and layout of widgets also follow natural and logical order.

- **User control and freedom**

The system is controlled and operated by user. User has the freedom of determining when to start the search task and when to stop. User can use the system as a simple PubMed search engine with basic operations like type queries, click on search button, change pages of the results and view results. User could also proceed addition tasks within the system in order to achieve a high recall and high precision search. User can click on “stop and export” button whenever he/she wants to quit from the current search task. The results of current search task will be downloaded automatically right after user clicks on the button which will reduce the concern about losing the results.

- **Consistency and standards**

Every widget appears on the user interface has its own use and uses different words. User does not need to worry about different words or actions mean the same thing.

- **Error prevention**

The user interface guides user to follow the workflow using the disabled attribute of each widget in order to prevent errors. User cannot click on any other buttons in the Operation Menu like “Submit”, “Start Training” before he/she clicks on “Start Innerloop” since those actions must be performed after the document data are inserted into database. User cannot click on “Start Innerloop” before he/she performs a search action. Once user has already started an iteration of inner-loop, the Query Form widget would be disabled which means user could not modify the query during the inner-loop. After user clicks the “Stop innerloop and start outerloop”, the Query Form will be activated to let user modify queries to increase recall in outer-loop.

- **Recognition rather than recall**

The system provides various instructions for user. When user hovers mouse on each button, a tooltip will show up in order to remind user about the function of each button.

Thus, user does not need to remember information from one action to another since the instructions are visible and are easy to retrieve.

- **Flexibility and efficiency of use**

Techniques including Ajax and accelerated storage of SQL increase the speed of responding, which allow user to take frequent actions.

- **Aesthetic and minimalist design**

The design of the user interface follows the principle of simple design, which does not contain any irrelevant or rarely needed information.

- **Help users recognize, diagnose, and recover from errors**

Error messages will be expressed in understandable plain language in the pop-up windows when user proceeds inappropriate actions. For example, when user types in a page number which exceed the page rage, a pop-up alert window will show up indicating that the user inputs an invalid page number.

- **Help and documentation**

Since user may not familiar with the double-loop mechanism, a user guidance is necessary. User can view the guidance page by clicking on the link under title of the main page. The guidance page also uses language which are easy to understand.

6 Conclusions and Limitations

The summary of prior works proofs that challenges and difficulties remain on performing systematic/literature review on online retrieval systems which aims at achieving high recall in addition to high precision, which addresses RQ1. In answer to RQ2, the produced system followed the ReQ-ReC framework using modular design, database design and interface design and implemented the double-loop mechanism and key functions required in chapter 1 successfully. Since the core search function of the system uses PubMed search API, it is reasonable to believe that the system could be embedded to PubMed, which answers RQ3. However, there still exists some limitations caused by time and effort limitations:

- Real demand analysis and usability evaluation test were not proceeded. In order to provide better search services to end users, I was supposed to take real demand analysis from end users, for example, a real interview with health science librarians whose main works are systematic reviews. In this way, I could design my system better based on their real demands. In addition, usability test should be executed in order to test if the user could use the system fluently with a professional documentation/guidance and whether they are satisfied with the search results after several iterations of the double-loop mechanism.
- The assumptions need to be further verified. When implementing the system, I made several assumptions which are mentioned in chapter 3. However, user might not be able to determine whether a document is relevant or not only using the

abstract and title of the document. User may need more information like methods, findings, and full-text of the article to make the judgement. In addition, the 1000 documents may or may not enough to constitute a document pool. We need to find such information through real interviews with end users.

- The choice of classification model in the document classifier module may not strong enough. Tentatively, I chose Naive Bayes as the classification model and TF-IDF vectors as feature representation for simplicity. However, better choices could be selected such as Linear Classifier, Support Vector Machine, Bagging Models, etc. Further test is needed in order to choose the model which performs the best.

7 Reference

[1] Holzinger, A. (2016). Interactive machine learning for health informatics: when do we need the human-in-the-loop?. Brain Informatics, 3(2), 119-131.

[2] Krause, J., Dasgupta, A., Swartz, J., Aphinyanaphongs, Y., & Bertini, E. (2017). A workflow for visual diagnostics of binary classifiers using instance-level explanations. arXiv preprint arXiv:1705.01968.

[3] Salton, G., & Buckley, C. (1990). Improving retrieval performance by relevance feedback. Journal of the American society for information science, 41(4), 288-297.

[4] Rui, Y., Huang, T. S., & Mehrotra, S. (1997, October). Content-based image retrieval with relevance feedback in MARS. In Image Processing, 1997. Proceedings., International Conference on (Vol. 2, pp. 815-818). IEEE.

[5] Cox, I. J., Miller, M. L., Omohundro, S. M., & Yianilos, P. N. (1996). Bayesian Relevance Feedback for Image Retrieval.

[6] Lu, Y., Hu, C., Zhu, X., Zhang, H., & Yang, Q. (2000, October). A unified framework for semantics and feature based relevance feedback in image retrieval systems. In Proceedings of the eighth ACM international conference on Multimedia (pp. 31-37). ACM.

[7] Rui, Y., Huang, T. S., & Chang, S. F. (1999). Image retrieval: Current techniques, promising directions, and open issues. Journal of visual communication and image representation, 10(1), 39-62.

[8] Song, J. J., Lee, W., & Afshar, J. (2017). An effective High Recall Retrieval method. Data & Knowledge Engineering.

[9] McNamee, P., Mayfield, J., Rowe, S. Y., Rowe, A. K., Jackson, H. L., & Baker, M. (2017, May). High Recall Text Classification for Public Health Systematic Review. In The Thirtieth International Flairs Conference.

[10] Song, J. J., & Lee, W. (2017). Relevance maximization for high-recall retrieval problem: finding all needles in a haystack. The Journal of Supercomputing, 1-24.

[11] Adeva, J. G., Atxa, J. P., Carrillo, M. U., & Zengotitabengoa, E. A. (2014). Automatic text classification to support systematic reviews in medicine. Expert Systems with Applications, 41(4), 1498-1508.

[12] Tsafnat, G., Glasziou, P., Choong, M. K., Dunn, A., Galgani, F., & Coiera, E. (2014). Systematic review automation technologies. Systematic reviews, 3(1), 74.

[13] Ananiadou, S., Rea, B., Okazaki, N., Procter, R., & Thomas, J. (2009). Supporting systematic reviews using text mining. Social Science Computer Review, 27(4), 509-523.

[14] Boeker, M., Vach, W., & Motschall, E. (2013). Google Scholar as replacement for systematic literature searches: good relative recall and precision are not enough. BMC medical research methodology, 13(1), 131.

[15] Li, C., Wang, Y., Resnick, P., & Mei, Q. (2014, July). Req-rec: High recall retrieval with query pooling and interactive classification. In Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval (pp. 163-172). ACM.

[16] https://www.ncbi.nlm.nih.gov/home/develop/api/

[17] https://scikit-learn.org/stable/

[18] https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/

Appendix

1. Development Environment

- Windows 10
- XAMPP (Apache + MySQL)
- Python 3.7

2. Technology Stack

- Client Side: JavaScript, JQuery, HTML, CSS
- Server Side: PHP, MySQL Database, Python (document classifier) scikit learn library

3. A list of program files

File Name	Use
dbconnect.php	Connect to MySQL database for further use.
get_abstracts.php	Retrieve abstract of an article from database and return to front-end.
get_update_after_train.php	Retrieve data from database after training finished and return to front-end.
getmax_taskid.php	Get latest task ID in order to create new task ID.
index.html	The front-end main web page of the system.

index-script.js	Perform actions for each widget on the main web page (index.html) using jQuery and Ajax.
insert_data.php	Insert first 1000 (or less than 1000) document data into database.
insert_query.php	Insert current query into database.
show_after_insert.php	Retrieve data from database after inserting finished and return to the front-end.
start_train.php	Call training.py to train data using exec().
stop_and_export.php	Stop current task and export results of current task.
styles.css	The style sheet of the front-end page.
training.py	Train data in current inner-loop and update data in the database.
update_labels.php	Update user labels into database.

Table 6: A list of program files and the use of them

4. SQL Scripts for creating table

```
create table articles
(
  taskid       int   not null,
  queryid      int   null,
  artid        int   not null,
  title        text  null,
  abstract     text  null,
  label        int   null,
  score        float null,
  pred_label   text  null,
  uncert_score float null,
  constraint articles_pk
    unique (taskid, artid)
);
```

Figure 15 SQL Script for creating table “articles”

```sql
create table queries
(
  taskid   int  null,
  queryid  int  not null,
  querystr text not null
);
```

Figure 16 SQL Script for creating table “queries”

5. User Interface Screenshots

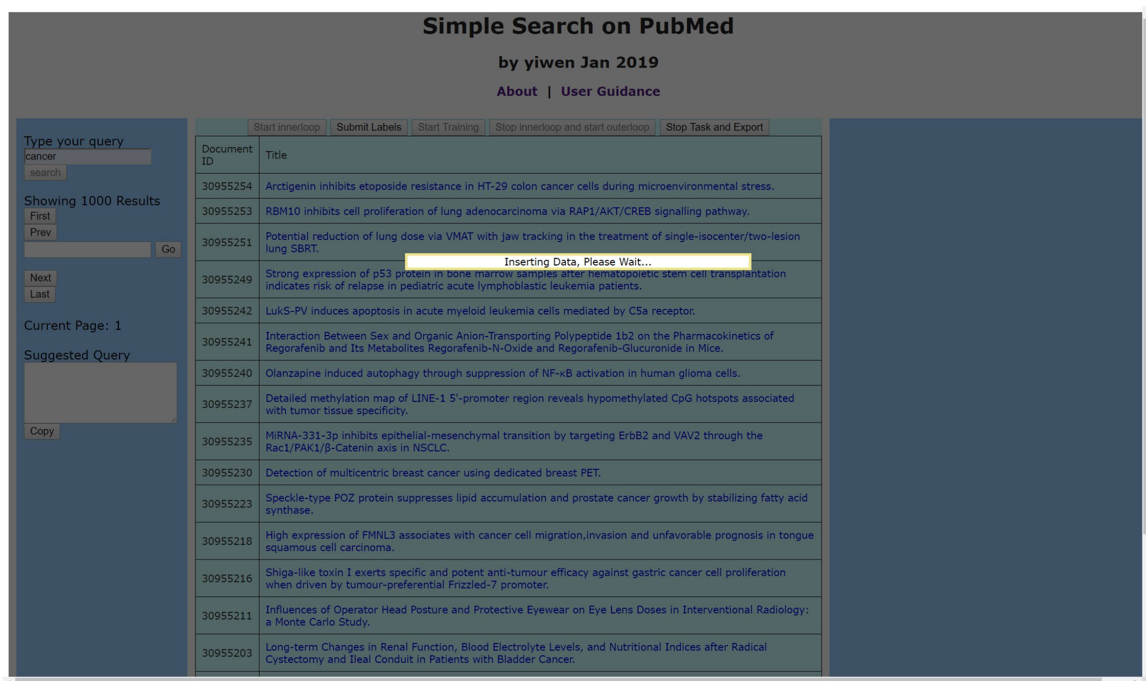


Figure 17 Block UI when inserting data

Figure 18 Drop down menus for user to label

Figure 19 Block UI when training data

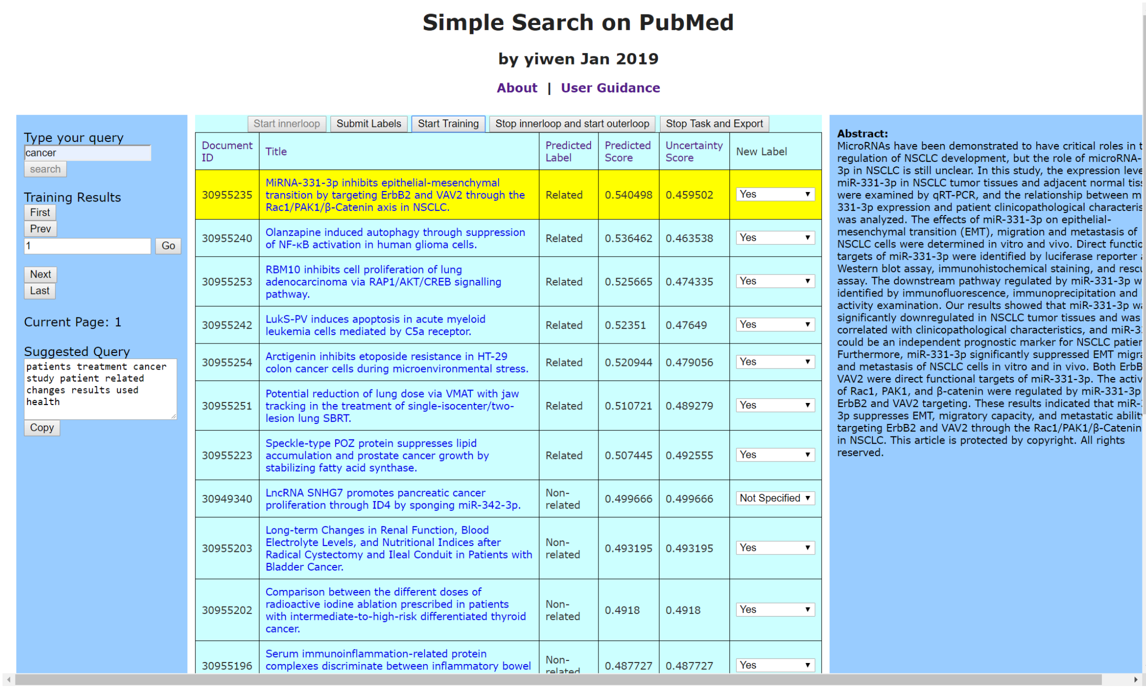


Figure 20 Show abstract when hover on a row

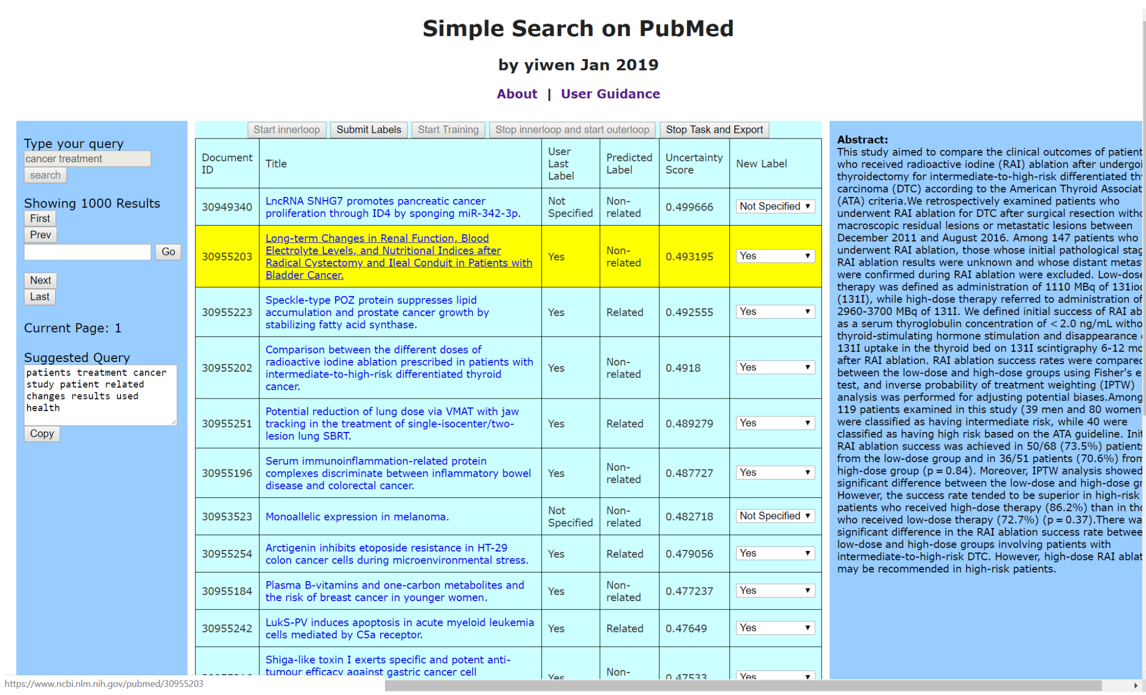
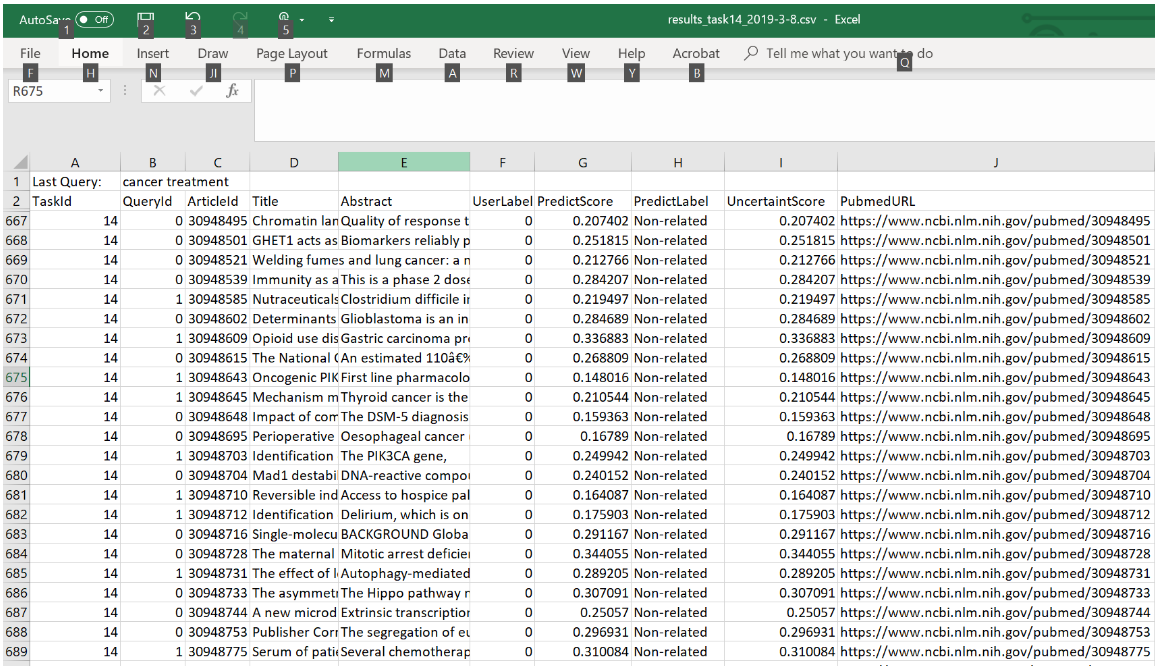


Figure 21 Use suggested query to start outer-loop



	A	B	C	D	E	F	G	H	I	J
1	Last Query:	cancer treatment								
2	TaskId	QueryId	ArticleId	Title	Abstract	UserLabel	PredictScore	PredictLabel	UncertaintScore	PubmedURL
667	14	0	30948495	Chromatin lan	Quality of response t	0	0.207402	Non-related	0.207402	https://www.ncbi.nlm.nih.gov/pubmed/30948495
668	14	0	30948501	GHET1 acts as	Biomarkers reliably p	0	0.251815	Non-related	0.251815	https://www.ncbi.nlm.nih.gov/pubmed/30948501
669	14	0	30948521	Welding fumes and lung cancer: a n		0	0.212766	Non-related	0.212766	https://www.ncbi.nlm.nih.gov/pubmed/30948521
670	14	0	30948539	Immunity as a	This is a phase 2 dose	0	0.284207	Non-related	0.284207	https://www.ncbi.nlm.nih.gov/pubmed/30948539
671	14	1	30948585	Nutraceuticals	Clostridium difficile ir	0	0.219497	Non-related	0.219497	https://www.ncbi.nlm.nih.gov/pubmed/30948585
672	14	0	30948602	Determinants	Glioblastoma is an in	0	0.284689	Non-related	0.284689	https://www.ncbi.nlm.nih.gov/pubmed/30948602
673	14	1	30948609	Opioid use dis	Gastric carcinoma pr	0	0.336883	Non-related	0.336883	https://www.ncbi.nlm.nih.gov/pubmed/30948609
674	14	0	30948615	The National (	An estimated 110â€%	0	0.268809	Non-related	0.268809	https://www.ncbi.nlm.nih.gov/pubmed/30948615
675	14	1	30948643	Oncogenic PIK	First line pharmacolo	0	0.148016	Non-related	0.148016	https://www.ncbi.nlm.nih.gov/pubmed/30948643
676	14	1	30948645	Mechanism m	Thyroid cancer is the	0	0.210544	Non-related	0.210544	https://www.ncbi.nlm.nih.gov/pubmed/30948645
677	14	0	30948648	Impact of com	The DSM-5 diagnosis	0	0.159363	Non-related	0.159363	https://www.ncbi.nlm.nih.gov/pubmed/30948648
678	14	0	30948695	Perioperative	Oesophageal cancer	0	0.16789	Non-related	0.16789	https://www.ncbi.nlm.nih.gov/pubmed/30948695
679	14	1	30948703	Identification	The PIK3CA gene,	0	0.249942	Non-related	0.249942	https://www.ncbi.nlm.nih.gov/pubmed/30948703
680	14	0	30948704	Mad1 destabi	DNA-reactive compo	0	0.240152	Non-related	0.240152	https://www.ncbi.nlm.nih.gov/pubmed/30948704
681	14	1	30948710	Reversible ind	Access to hospice pal	0	0.164087	Non-related	0.164087	https://www.ncbi.nlm.nih.gov/pubmed/30948710
682	14	1	30948712	Identification	Delirium, which is on	0	0.175903	Non-related	0.175903	https://www.ncbi.nlm.nih.gov/pubmed/30948712
683	14	0	30948716	Single-molecu	BACKGROUND Globa	0	0.291167	Non-related	0.291167	https://www.ncbi.nlm.nih.gov/pubmed/30948716
684	14	0	30948728	The maternal	Mitotic arrest deficie	0	0.344055	Non-related	0.344055	https://www.ncbi.nlm.nih.gov/pubmed/30948728
685	14	1	30948731	The effect of l	Autophagy-mediated	0	0.289205	Non-related	0.289205	https://www.ncbi.nlm.nih.gov/pubmed/30948731
686	14	0	30948733	The asymmetr	The Hippo pathway r	0	0.307091	Non-related	0.307091	https://www.ncbi.nlm.nih.gov/pubmed/30948733
687	14	0	30948744	A new microd	Extrinsic transcriptior	0	0.25057	Non-related	0.25057	https://www.ncbi.nlm.nih.gov/pubmed/30948744
688	14	0	30948753	Publisher Corr	The segregation of eu	0	0.296931	Non-related	0.296931	https://www.ncbi.nlm.nih.gov/pubmed/30948753
689	14	1	30948775	Serum of pati	Several chemotherap	0	0.310084	Non-related	0.310084	https://www.ncbi.nlm.nih.gov/pubmed/30948775

Figure 22 Sample of exported file

6. Github Link

https://github.com/yiwen9586/master-project