UNIVERSITY OF LIVERPOOL

GROUP PROJECT DECLARATION OF ACADEMIC INTEGRITY

NAME (Print)	Zeyu Liao
STUDENT NUMBER	201298327
GROUP NAME	COMP214 groups 6
MODULE TITLE/CODE	201718-COMP214 - AI GROUP PROJECT
TITLE OF WORK	Precision Search Filter

This form should be completed by one student on behalf of the group and appended to any piece of work that is submitted for summative assessment. Submission of the form by electronic means by a student constitutes their confirmation of the terms of the declaration.

Students should familiarise themselves with Section 9 of the Code of Practice on Assessment and Appendix L of the University's Code of Practice on Assessment which provide the definitions of academic malpractice and the policies and procedures that apply to the investigation of alleged incidents.

Students found to have committed academic malpractice are liable to receive a mark of zero for the assessment or the module concerned. Unfair and dishonest academic practice will attract more severe penalties, including possible suspension or termination of studies.

STUDENT DECLARATION

I confirm that the group has read and understood the University's Academic Integrity Policy.

I confirm that the group has acted honestly, ethically and professionally in conduct leading to assessment for the programme of study.

I confirm that no member of the group has copied material from another source nor committed plagiarism nor fabricated data when completing the attached piece of work.

I confirm that no member of the group has previously presented the work or part thereof for assessment for another University of Liverpool module.

I confirm that no member of the group has copied material from another source, nor colluded with any other student in the preparation and production of this work.

I confirm that no member of the group has incorporated into this assignment material that has been submitted by any other person in support of a successful application for a degree of this or any other University or degree awarding body.

SIGNATURE……Zeyu Liao……

DATE……09/05/2018……

Precise Search Filter Report

COMP 214 GROUP 6

ZEYU LIAO, YAN MO, TIANHUI ZHANG,
HAOMIN SONG, KUN WANG, MING CHENG

1. TEAM MEMBER SUMMARY

Group Member	Major Tasks	Additional Contribution
Zeyu Liao	• Leader • Requirement elicitation and analysis • System structure design and implementation • Acceptance/ user testing • Linux server setup and domain purchase • Module integration	• Come up with the human process idea and provide a system prototype • Project planning and management • Server maintenance • raw data manipulation and source code maintenance
Yan Mo	• Requirement non-functional part design • Algorithm documentation design • Algorithm documentation implementation • Modify all of the design report • Evaluation and future work design • Modify part of the portfolio	• Data structure design • Data structure implementation • Algorithm design with Liao Zeyu • Algorithm design implementation • Performance testing
Tianhui Zhang	• Design database • Create the frame of web page • Design and implement keyword extraction algorithm • Connect the web page to the python script with Django in server • Write system testing	• Write corresponding part of each document. • Help Zeyu Liao modify the testing document • Help Yan Mo, Zeyu Liao to debug the system • Gave some idea when designed algorithm
Kun Wang	• Plan for the project and draw Gantt chart • Requirement analysis • Modify requirement specification • Modify design report • Write user manual • Write portfolio	• Record meeting notes • Book meeting room • Help Yan Mo find reference
Ming Cheng	• Write the introduction of requirement specification • Write the background in design • Write user manual	• Modify the grammar errors in the reports • Finding background information related to

	• Write portfolio • Create tables in database • Modify the web page of the system	NLP and Word2Vector • Requirement analysis • Document typesetting
Haomin Song	• Design stage plan • User interface design • Promote performance and eliminate errors of web crawler • BCS principles analysis • Portfolio writing and collation • Bibliography collation	• Interaction diagram design • User interface state diagram design • Boundary diagram design • Use case diagram design

2. OVERVIEW

2.1 Application Mission

Precision Search Filter is a web-based search engine. The website address is: http://www.trackai.tech. It aims to display a short text-form paragraph for questions queried by users. This paragraph, extracted from the best websites selected, is the most relevant answer to the query. Therefore, users do not need to click links to select the information they need from a huge number of websites, which can save them a lot of time. In addition, there is no data noise and advertisement in the webpage. As a result, there is no interference when users search.

2.2 Usage Instruction

The framework of web application is similar to normal search engines, such as Google, Bing or Yahoo, which makes it more familiar to the users. User can click the search bar on the center of the webpage and enter the query. When clicking the “Search” button, the system will analyze the query and show the original query and the best answer. Besides, the reference link of this answer is displayed under the answer. Those who want to refer to the original website of the answer can click the link and the browser will redirect to the reference website.

2.3 Target Users

Because of its convenience, the target users of this system are the entire public, especially who want to have a quick search experience and get a concise text results. This kind of user mainly comes from two groups:

- Urgent User: People who lack of patience but have experience in searching and prefer short text.
- Inexperienced User: People who cannot distinguish information and feel difficult to locate the desired information.

Both two groups of users need quick and concise search without wasting time. In this case, they can benefit from the Precision Search Filter.

3. ACHIEVEMENT

3.1 Product Achievement

The system is divided into two parts: the web user interface and the server. As soon as users type query in the website, the query message will be sent to the server. The server will first extract keywords from the query and check database. Then it selects the best website and best paragraph from a list of websites crawled from Google. Finally, the result will be stored in the database and passed to the user.

3.2 Functional Requirement Satisfied

- **Keyword extraction**

  The most relevant keywords are extracted from the query. It is used to search for answers to the questions and then stored as attributes in the database.

- **Page crawling**

  The top 20 links with the title and content are crawled from Google website [1] [2]. These links are most related to the question asked by users. Therefore, the answer is

believed to exist in these links. All crawled data are stored in the temporary directory for further analysis.

- **Google Featured Snippet Extraction**

  Google Featured Snippet is the most possible area that contains the answer. If existing, the system firstly extracts information in Google Featured Snippet and returns it as the result.

- **Best website selection**

  Scoring system is design to focus on user queries to select the best web page and the feedback performance is quite reliable. We also quantified the feature words extracted from the text to represent the text and it is well recognized by the computer.

- **Best paragraph generation**

  We have completed the analysis of the text. Although it is based on the text structure and key fields, the most appropriate paragraph has been selected and returned to the user.

- **Cosine similarity implementation**

  Cosine similarity is used to calculate the similarity between two word vectors. This method is used in both get best paragraph and the best link [3].

- **Bayes Classification**

  It divides the questions into two categories. “What, When, Who” & “how, why”. The classification is useful when dealing with the question analysis [4].

- **Database Implementation**

  A database is designed to store answers along with keywords in order to return the answer more quickly when other users ask the similar questions that had been asked before [5] [6].

- **User Interface implementation**

  User can input queries in the question bar and view the answer in the answer box. The original link where the answer comes from is also provided as a reference to users in case they need further search.

3.3 Non-functional Requirement Satisfied

- The system is easy to use without taking any training before using the system.
- The system is evolvable for future development.
- The result is limited under 1000 characters.
- The searching process will not be more than 6 seconds.
- The system is a web application. Any user can use it without signing up.

4. EVALUATION

4.1 Strengths

Four main strengths are demonstrated as below:

- This system is a technical breakthrough. It implements the technologies include generate the best link and generate the best paragraph.
- This system has enormous commercial potential. All the operations are automatically done by the system and the system guarantees to return the best URL if there exists one.
- This system is extensible. It is easy to add features. For example, there is a database to store the question and answer. With the accumulation of the database, the system will response faster and will be more accurate.
- This system has an outstanding performance on selecting the best link. The selecting process simulates the working process of human behavior when selecting webpages based on the abstract and the title of the webpage.

4.2 Weaknesses

- Multi-threaded programs are currently unstable. Due to incomplete compatibility between Django framework and python script, the best paragraph generated by the system might be overwrite by the previous query. The answer returned to the current user might be the answer that should be allocated to the other user who asked before.
- The database in the system is small in the current stage, which will lead to spending more time on returning the answer to users. According to the test documentation, returning an

answer which is already in the database takes 500ms. However, returning an answer that is not in the database takes around 5s, which is far more than the situation when answers are in the database.

- The analysis of the semantics and the understanding of the user queries are not implemented in the system. For example, “how” and “how old” are different parts of speech but they are not well analyzed and differentiated when extracting keywords from the users’ query.

5. FUTURE DEVELOPMENT

There are numerous potential development possibilities that can be implemented in the future.

- **Update database**

  The database should be more completed so that the algorithm can be promoted with the accumulation of database. For the answers to some questions, we can use tricky method of querying the database to find the answers quickly.

- **Design a more competing ranking system**

  It is recommended to design a much more precise ranking algorithm to rank each paragraph according to the training and analysis of semantics based on NLP.

- **Analyze the semantic**

  The answer returned by the system currently is the original paragraph of the answer website, which might contain useless information. In order to provide users with more precise useful information, the semantics should be analyzed and the answer should be a text generated by the designed algorithm.

- **Add voice-text conversion and text-voice conversion functions**

  In order to serve more special groups of people such as people who do not know how to type or who are blind, the functions of voice-text and text-voice conversion will be helpful.

6. SATISFACTION OF THE BCS CODES OF PRACTICE AND CONDUCT

According to the British Computer Society (BCS), all members and products of this project have responsibility to meet the professional standards and rules. This section will justify in three aspects: public interest, professional competence and integrity, duty to relevant authority and professionalism. All aspects are main parts of the BCS code of conducts.

6.1 Public interest

This precision search filter follows the public interest, including public privacy, legitimate rights of third party and activities without discrimination.

In terms of public privacy, no private information of users is needed for this search filter. All users are accessible to use this web-based search filter directly. In addition, information stored in the database is keywords of query and answers extracted from open source websites, rather than personal information. As a result, no public privacy is invaded.

This search filter is based on the Google search engine, which is regarded as the third party. According to the Robots Exclusion Standard (RES), the web crawler has to obey the standards and permission of Google. Therefore, this search filter follows the rules in robot.txt of Google, which identifies areas that Google permits to visit [7]. It only crawls the open source webpages that Google allows. Moreover, the purpose of this filter is to promote efficiency to search for information and extract answers from other websites. To avoid infringement, the system provides reference links about the origins of answers. User can refer to original links for further reading as well.

On the other hand, this search filter has no prejudice to any group of people, regardless of sex, nationality, race and ethnic origin. As a result, this project fulfills the public interest.

6.2 Professional competence and integrity

This search filter provides suggested answers. All work ranging from building user interface, server to database comes from efforts of group members. We confirm that all answers are extracted from existing websites, however, cannot guarantee the absolute accuracy of answers. In addition, national language process is not included in this system. Therefore, the query that needs linguistic analysis cannot get accurate answers.

6.3 Duty to relevant authority and profession

The entire project process is under the supervision of University of Liverpool. All knowledge is used to provide functionality and promote efficiency of this project. In addition, instruction and performance can be referred in the User manual. There is no information misrepresented or withheld. In this case, it corresponds to the BCS codes of Practice and Conduct.

7. BIBLIOGRAPHY

[1] Dataquest, (2018). *Python Web Scraping Tutorial using BeautifulSoup*. Available from: https://www.dataquest.io/blog/web-scraping-tutorial-python/ (08 May 2018).

[2] Stack Overflow, (2018). *Scrape google search snippet results*. Available from: https://stackoverflow.com/questions/45555330/scrape-google-search-snippet-results (08 May 2018).

[3] Masongallo.github.io, (2018). *Implementing and Understanding Cosine Similarity.* Available from: https://masongallo.github.io/machine/learning,/ python/2016/07/29/cosine-similarity.html (08 May 2018).

[4] Jurafsky, D., and Martin, J., (2014), *Speech and language processing*. [India]: Dorling Kindersley Pvt, Ltd. pp. 6.1-6.19.

[5] Connolly, T.M., and Begg, C.E., (2005). *Database systems: a practical approach to design, implementation, and management*. Pearson Education.

[6] Chin, F. Y., Ozsoyoglu, G., (1981), Statistical database design[J]. *ACM Transactions on Database Systems*, 6(1):113-139.

[7] Google co., (2018), *Robots Exclusion Standard*. Available from: https://www.google.com/robots.txt (08 May 2018).

[8] NLTK Team, (2018). *Natural Language Toolkit*. Available from: http://www.nltk.org/(08 May 2018).

[9] Condor.cc.ku.edu, (2018), *Lynx Users Guide v2.3*. Available from: http://condor.cc.ku.edu/~grobe/docs/Lynx_users_guide.html (08 May 2018).

COMP214 Design Report

Precision Search Filter

Group 6

Abstract

The Precision search filter is software for answering users' queries. A brief answer will be provided directly on the screen. In this report, project description will be given first. Then, use cases, algorithm design, server design, database design and testing design will be demonstrated in detail. In the end, a review of the project process will be given.

Contents

1 PROJECT DESCTIPTION

1.1 BACKGROUND

As the fast development of the Internet, the website has become indispensable to human beings, especially in retrieving data and information. According to one web statistics organization called Netcraft in Bath, England, there are 625 million active websites on the Internet as of the time of publication [1]. The efficient way to find answers in a number of websites is searching. Using searching makes it easy and convenient to know more knowledge about the world. Our project is created based on this situation but attempt to apply a more convenient way to convert result to users. Considering the existing website search engine, such as Google or Bing, can only give recommended links after searching, which also needs a lot of time to scan these links with their abstracts and read the context in it. We want to use AI knowledge to help people to abbreviate searching process.

In order to achieve such process, as planned in the requirement phase, our "Precision Search Filter" is a quick and succinct search, in the form of a website with an input box and served to find the search result. The search result will be generated by clicking "search" button. The operation is as easy as general search engine. The whole system can be executed through five parts. First, query will be input. Then, the input will be transferred into server. When server receives the input, it will recognize it and crawl the relevant links of google. After that, top 20 searched websites will be chosen, compared and merged by using AI knowledge. Finally, the exact answer generated by the system will be output. This report will lay emphasis on the algorithm design, server design, database, interface design and also an introduction to testing design.

1.2 OBJECTIVES

The main purpose of our project is to receive the input from the user, select, extract the information from google results, and to generate the direct short text with words limit. Main role of our product is for informative search. Designed algorithm is core for the anticipated performance of our system. Additionally, a database system will also be needed for storing data and quick search.

The intended audience of the Precision Search Engine system is people who want to have a quick search on a webpage and a concise text search results. These target users can be divided into two groups:

1. Experienced User: People who have experience in searching and prefer short text.
2. Naïve User: People who do not have the search skills, cannot use the mobile phone, or even not able to type the message, see and read the content.

Our product providing such search will save a great deal of time for users, especially when doing a Q&A competition about general knowledge. It can also be used in education and many other domains where needed search.

1.3 ORIGINAL PROPOSAL AND CHANGES

There are three main differences between the project in the design report and in the requirement report.

Contrary to the original design which plans to produce a maximum 140 words' fluent sentence that is generated based on the keyword, the final design is going to output either phrases or sentences extracted straightforwardly from the webpage because of the technology limit. In addition, the original link where the answer extracted from will be provided under the text output. It is harder to implement an algorithm to generate a complete sentence based on keyword extracted than what can be imagined, thus, the change is decided to make the project easier to realize.

In addition, the final design differs slightly from the original design in terms of the knowledge used. The original design intends to utilize web crawling, keyword extraction, natural processing language, and similarity comparing and website setup. The final design will add the classification algorithm because applying specific regular expression for each category is able to achieve a more precise output.

Although this was not in the original design, there will be a database added because we want to store the keywords and classification of the questions entered by users to offer them answers immediately if the questions have been searched before.

1.4 RESEARCHES

The searches we have done so far are shown as below.

- Algorithm: classification algorithm, cosine similarity and corresponding pseudo code.
- Database: Data dictionary, Entity-relation diagram
- User-interface (UI): User Website Map Diagram
- Applications: Lynx, mysql5.5, python3, and webpage HTML
- API: crawl webpage

2 PROJECT DESIGN

2.1 BACKGROUND

2.1.1 System Description

System of our project is presented in the form of website. After the user enters the website, the user can start to search by input text. The query of the user will be sent to project server. No constraints on query but several suggestions on query are included in User Manual, guiding for better searching performance.

Firstly, the keywords in the query will be extracted. Second, the query will be processed by classification algorithm to generate "tags" first, and then compare to the local database to

check whether suitable answer exists. Third, the query will be processed to google in order to crawl enough data. In the end, the best answer will be extracted using our algorithm and presented as a short text back to the user.

2.1.2 Functional Description

- Find the best answer according to the query.

2.1.3 Use Cases

2.1.4

The use case diagram is shown in Figure 1

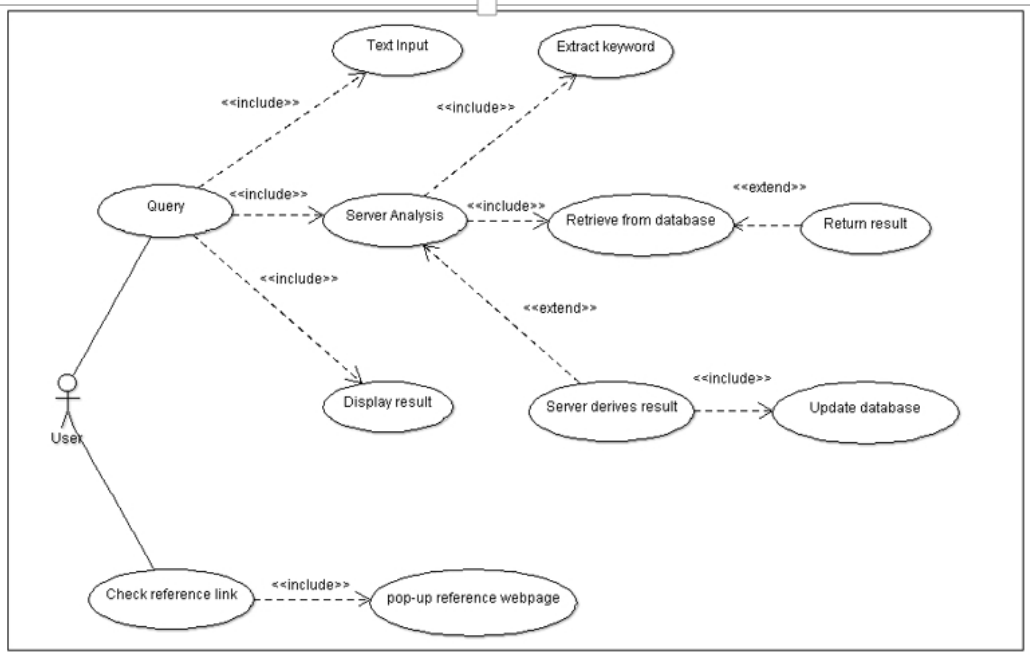


Figure 1: Use case Diagram

ID	UC1
Name	Query
Description	Raise the query to find answer
Pre-condition	Server Process must be in service
Event flow	1. UC2 Text Input 2. UC3 Server Analysis 3. UC4 Display result
Post-condition	Message send to Server
Includes	UC2 Text Input

	UC3 Server Analysis UC4 Display result
Extensions	None
Trigger	User Click Search Button

Table 1: UC1

ID	UC2
Name	Text Input
Description	Get user's text input
Pre-condition	Server process must be active
Event flow	1. user click text box 2. user input text 3. user click send
Post-condition	Text input stored
Includes	None
Extensions	None
Trigger	User click text box

Table 2: UC2

ID	UC3
Name	Server Analysis
Description	Analyse user's query
Pre-condition	Server Process must be in service Input done
Event flow	1.Check User Input 2.Use algorithm to get the answer
Post-condition	The answer are presented
Includes	None
Extensions	None

Trigger	User click end button and input is invalid

Table 3: UC3

ID	UC4
Name	Display result
Description	Show the result to the user
Pre-condition	Best answer has been generated
Event flow	Return the answer to the user
Post-condition	User get the information
Includes	None
Extensions	None
Trigger	Querying are done

Table 4: UC4

ID	UC5
Name	Extract Keyword
Description	Extract keyword from decrypt message
Pre-condition	Server process must be active
Event flow	1. use algorithm to extract keyword from received message 2. store the keyword
Post-condition	The keyword extracted and stored
Includes	None
Extensions	None
Trigger	Server received query and do the analysis

Table 5: UC5

ID	UC6
Name	Retrieve from database
Description	Check if the result already in database
Pre-condition	Server Process must be active
Event flow	1. receive message from client 3. check database

Post-condition	Return result if found result in database
Includes	None
Extensions	UC7 Return result
Trigger	Receive message from client

Table 6: UC6

ID	UC7
Name	Return Result
Description	Return the answer from database
Pre-condition	Server process must be active
Event flow	Return the result
Post-condition	Result is ready to display
Includes	None
Extensions	None
Trigger	Result is generated

Table 7: UC7

ID	UC8
Name	Server drives result
Description	Get the result form server
Pre-condition	Server service must be active Receive encrypt Message from Client
Event flow	1. Use algorithm get result
Post-condition	Result generated
Includes	UC9 update database
Extensions	None
Trigger	Query has been sent to server

Table 8: UC8

ID	UC9
Name	Update database
Description	Store result to database

Pre-condition	Data not found in database, webpage generated
Event flow	1. load generated webpage 2. stored to database
Post-condition	A new record add to database
Includes	None
Extensions	None
Trigger	Record stored in database

Table 9: UC9

ID	UC10
Name	Check reference link
Description	Click the link
Pre-condition	Result generated
Event flow	1. Click the link of webpage 2. scan the context
Post-condition	Reference link generated
Includes	None
Extensions	None
Trigger	Server analyse query

Table 10: UC10

ID	UC11
Name	Pop-up reference webpage
Description	Pop-up the reference webpage
Pre-condition	Server Process must be active
Event flow	1. open the webpage
Post-condition	Webpage appeared
Includes	None
Extensions	None
Trigger	Click the link

Table 11: UC11

2.2 ALGORITHM DESIGN

2.2.1 Background

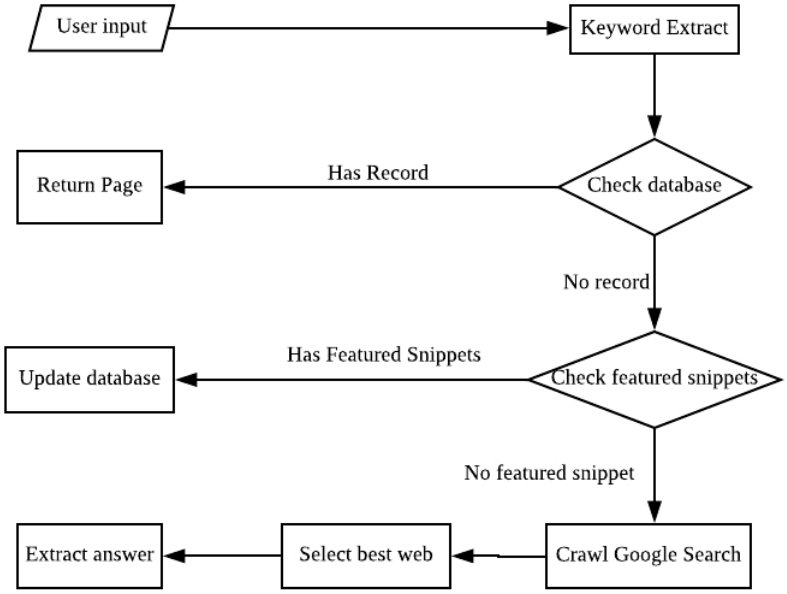


Figure 2: Process Diagram of system

The diagram above demonstrates the process of precision search engine system. Our algorithm focuses on finding the best website and suitable answer.

To find a best website, first we will divide queries into two genres. Genre One is about “what”, “who”, “when”. Genre Two is about “how &why”. If query belongs to the first genre, text similarity between query and title should be considered mainly and then the similarity of abstract and query will be considered. If query belongs to the second genre, text similarity between query and abstract should be considered mainly and then the similarity of title and query will be considered.

To find a best answer, an ad-hoc scoring system has been designed. It will give each text paragraph a relevant score, which demonstrates whether paragraph is suitable. Detailed introduction is in the Algorithm specification.

In general, there are several common algorithms to implement text classifications such as naive Bayes, convolutional neural network, recurrent neural network and so on. Since the limit of learning queries resource, the final applied algorithm is naïve Bayes sentence classification. The classification method a statistical classification method based on Bayes theorem and independent assumption of characteristic condition. For a given input, Bayes theorem is used to find the maximum output of the posterior probability. It has been proved that the model achieves good classification performance across a range of sentence classification. Also, it is very convenient for implementation.

2.2.2 Algorithm Specification

- **Naïve Bayes Text classification model**

Basically, Naïve Bayes Test classification model is to classify queries into two classes. Firstly, we need to create a word list for each class. It is not necessary to collect all the words used during classification. Naïve Bayes model can have a good performance using only part of featured words. The second step is to digitize the sentence. Actually, this model provides a Bernoulli model, which is also called word set pattern, to implement digitization. Finally, Bayes' rule is presented in the following, it gives us an approach to calculate conditional probability:

$$p(c \mid x) = \frac{p(x \mid c)p(c)}{p(x)}$$

Assuming w as the word set, $c1$ as the classification one and $c2$ as the classification two, we can predict the more suitable class query vector w belongs to:

$$MAX\left(p(c1 \mid w), p(c2 \mid w)\right)$$

- **Scoring method for the best paragraph based on cosine similarity**

For each paragraph, assuming $v1$ as the key words set and $v2$ as the synonyms set of key words, the amount of each key word that have been occurred will be recorded and then a score based on word frequency is generated, $S(v1)$. After that, the amount of each synonym word that have been occurred will be recorded and then a score, $S(v2)$, based on word frequency is generated as well. Finally, the similarity of paragraph keyword and query keyword will be calculated and get$S3$.Three scores are summed up to a weighted mean. The final score is to measure how suitable the answer is for the best answer.

- **Cosine Similarity**

Cosine similarity algorithm is applied to compute the similarity between two texts. It is a between two non-zero vectors. To use it in this system, firstly, we get all the distinct meaningful words. Then term frequency vectors of keywords are formed. For example, s0 = "This is sentence example sentence one" and s1= "This example sentence has six words". After transformation, two vectors are acquired: "This example sentence has six words". Therefore, the term frequency vectors are formed. After such operations, the problem is transferred into a standard mathematical problem. Given two vectors of attributes, A and B, the cosine similarity is represented as:

$$similarity = \cos(\theta) = \frac{A \bullet B}{\| A \| \| B \|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

2.2.3 Pseudo Code

```
Function  main( Query, userID)
   Set workDir = "/var/www/project/finalCode/"
   Set returnlist=[]
   Set URL = Query.replace(" ","+")
        URL = "https://www.google.com/search\?num\=30\&q\=" +URL
   Set tempDir = "/var/www/project/temp"
        tempDir = tempDir + str(userID) + "/"

   Set featuredSnippet = download_snippets.run(Query)
   Print (featuredSnippet)
   Print ("Downloading Google Results....")
   Print ("lynx -dump "+ URL + " >"+ tempDir + "gone.tmp")

   #method to store keyword in keywordResult
   Set keyword_Result = keywordExtract.run(Query)
   Print (keyword_Result);
   Set bestPageURL= bestLink.run(Query , tempDir )
   Print ("Best Link " + bestPageURL)

   Print ("Downloading Best Page....")
   pageCrawler.run(bestPageURL, tempDir, "target.html")

   print ("Generateing paragraph")
   os.system("bash generateParagraph " + tempDir)

   Set paragraphNumber = os.popen("bash getParagraphNo " + tempDir).read().strip();
   Print (paragraphNumber)

   Set paragraphs=loadParagraph.run(tempDir, paragraphNumber)
   Set paragraph_vector=paragraph_to_vector.run(paragraphs)

   Set best_para=best_answer.run(paragraph_vector,keyword_Result,Query,paragraphs)

   returnlist.append(best_para)
   returnlist.append(bestPageURL)
   if os.path.exists(tempDir):
      shutil.rmtree(tempDir)

   return returnlist
End function

Function getKeyword (String query)
  Import nltk
  Set list<keyword>=toolkit.extract(query)
```

```
  Return list<keyword>
End function

Function getBestLink(query, DIR):
   Set rawdata = []
   print (DIR)
   Set x=0 y=0
   while x<10:
      titleNo=os.popen("bash getTitleNo "+ DIR +" "+ str(y)).read().strip();
      title=os.popen("bash getTitle "+ DIR +" " + str(titleNo)).read().strip();
      abstract=os.popen("bash getAbstract "+ DIR +" " + str(titleNo)).read().strip();
      link=os.popen("bash getLink "+ DIR + " "+ str(titleNo)).read().strip();
      #paragraph=os.popen("bash getPara" + str(para)).read().strip();
      rawdata.append([]);
      y=y+1
      IF ("www.youtube.com" not  in link) and (".pdf" not in link)and("images?" not in link):
         rawdata[x].append(titleNo)
         rawdata[x].append(title)
         rawdata[x].append(abstract)
         rawdata[x].append(link)
         x=x+1
       End If
     End While
   query_vector=query.split()
   result=bayes.testingNB(query_vector)
   if (result==0):
     best_title=cosine_similarity.best_title_0(titles,query)
     for x in range(1,11):
         if(best_title==rawdata[x-1][1]):
           link=rawdata[x-1][3]

   else:
     best_content=cosine_similarity.best_title_1(titles,query,contents);
     for x in range(1,11):
         if(best_content==rawdata[x-1][2]):
           link=rawdata[x-1][3]
           print("Best title: ",rawdata[x-1][1])
   return link
End Function


Function get_best_paragraph(paras,keywords,syn_list,paragraphs):
   Set score_list=[]
```

```
    if len(paras)==0:
        Set result="No suitable paragraphs available.\n You can visit the following link directly"
    else:
        for index in range(len(paras)):
            p_vector=paras[index][1]
            temp_scores=score(p_vector,keywords,syn_list)
            score_list.append([paras[index],temp_scores])
        list4=sorted(score_list, key = operator.itemgetter(1), reverse=True)
        index_P=list4[0][0]
        Set result=paragraphs[index_P[0]
        print (result)
    return (result)
End Function

Function score(p_vector,keywords,syn_list):
    Set whole_word=keywords+syn_list
    Set score=0
    Set c_keywords=cosine_similarity.vector_to_counter(keywords)
    Set c_syn_list=cosine_similarity.vector_to_counter(syn_list)
    for i in range(len(p_vector)):
        for j in range(len(whole_word)):
            if (whole_word[j].lower()==p_vector[i].lower()):
                score=score+10+(len(p_vector)+1-i)*2
                c_para=cosine_similarity.vector_to_counter(p_vector)
                c1=cosine_similarity.get_cosine(c_para,c_keywords)
                c2=cosine_similarity.get_cosine(c_para,c_syn_list)
                score=score+int(c1*100+c2*100)
    return score
End Function

Function setOfWords2Vec(vocabList, inputSet):
    returnVec = [0] * len(vocabList)
    for word in inputSet:
        if word in vocabList:
            returnVec[vocabList.index(word)] = 1

    return returnVec
End Function

Function trainNB0(trainMatrix, trainCategory):
    numTrainDocs = len(trainMatrix)
    numWords = len(trainMatrix[0])
    pAbusive = sum(trainCategory) / float(numTrainDocs)
```

```
    p0Num = ones(numWords)
    p1Num = ones(numWords)
    p0Denom = 2
    p1Denom = 2
    for i in range(numTrainDocs)
        if trainCategory[i] == 1:
            p1Num += trainMatrix[i]
            p1Denom += sum(trainMatrix[i])
        else
            p0Num += trainMatrix[i]
            p0Denom += sum(trainMatrix[i])
    p1Vect = log(p1Num / p1Denom)  #use log function
    p0Vect = log(p0Num / p0Denom)
    return p0Vect, p1Vect, pAbusive

Function classifyNB(vec2Classify,p0Vec,p1Vec,pClass1): #compare the probability
    p1 = sum(vec2Classify*p1Vec)+log(pClass1)
    p0 = sum(vec2Classify*p0Vec)+log(1-pClass1)
    if  p1>p0:
        return 1
    else:
        return 0

Function testingNB(vector):
    listOPosts,listClasses = loadDataSet()
    myVocabList = createVocabList(listOPosts)
    trainMat=[]
    for postinDoc in listOPosts:
        trainMat.append(setOfWords2Vec(myVocabList, postinDoc))
    p0V,p1V,pAb = trainNB0(array(trainMat),array(listClasses))
    thisDoc = array(setOfWords2Vec(myVocabList, vector))
    result = classifyNB(thisDoc,p0V,p1V,pAb)
    return result
    End Function
```

2.3 SERVER DESIGN

2.3.1 Background

Server is playing a role to support the entire system, managing each part and making a response to user's requests. Hence, it is necessary in order to build a robust server to handle a large amount of requests.

The most common pattern to achieve the flexibility and robust is Model-View-Controller (MVC) framework, which can separate components of the whole system as well as allow developers work in parallel.

2.3.2 Structure and Framework

The ConcretController layer represents the controller that executes the event of ConcretView layer, and then update Model. It mainly contains three parts, extracts keywords from the user query, classifies the type of user query, as well as download relative article from Google.

The ConcretView Layer will transit user action to the controller, and update according to the model layer.

The ConcretModel layer will update the view layer according to the command generated by the controller layer. The model is a regular expression updated by the type of content, and applied one of the most relative webpage generated by Semantic Similarity Compare.

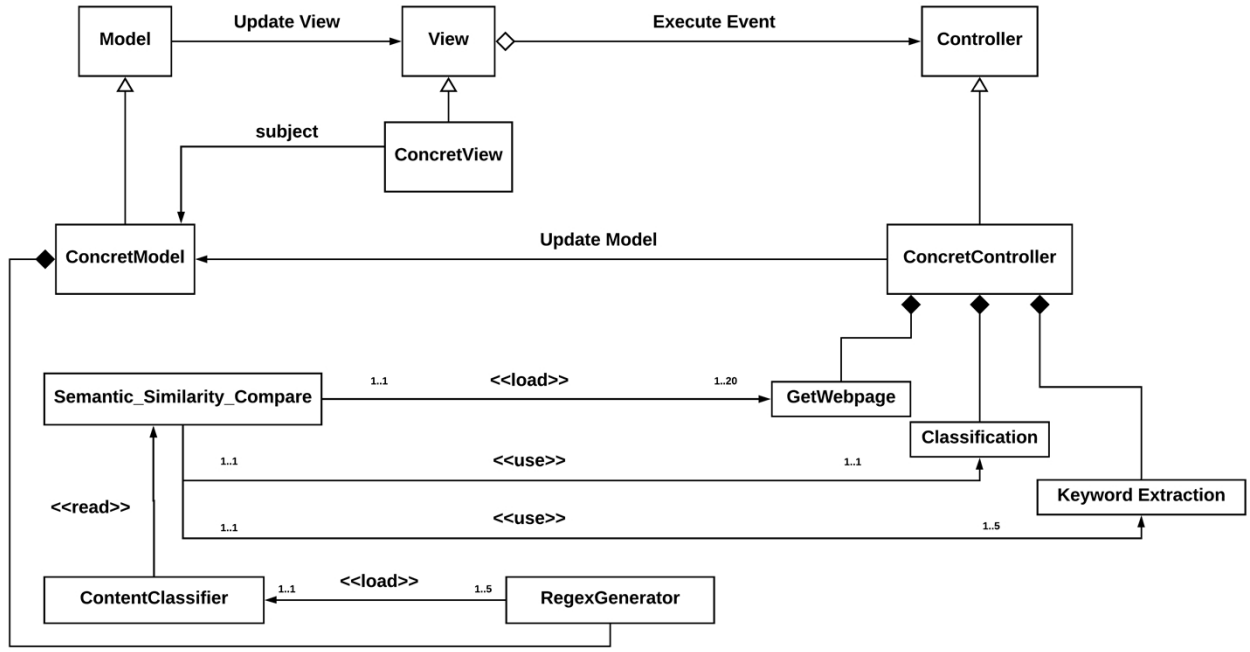


Figure 3: System structure

2.3.3 Interface Design

Interface	List<Keyword> getKeyword(String query)
Description	Extract keyword from user query
Parameters	Query – the user input

Return Value	The keyword of the query

Interface	List<URL> getWebpage(String query)
Description	Crawl relative webpage from google
Parameters	query – user input
Return Value	A list of relative webpage

Interface	String classification(String query, List <keyword> keyword)
Description	Classify the user query
Parameters	query-- user input keyword – the keyword extract from query
Return Value	A String that identify the query type

Interface	Webpage getBestLink(List<Webpage> page, String tag, List <keyword> keyword)
Description	Extract the most relative webpage
Parameters	Page-- a list of relative webpages keyword – a list of keywords tag – the type of query
Return Value	A webpage that best match the user query

Interface	List<paragraph> ContentExtract(Webpage)
Description	Extract different part of the webpage
Parameters	Webpage – the most relative webpage generated
Return Value	A list of paragraph

Interface	List<String> regexGenerator(List< paragraph> content, List<keyword> keyword,List<Synonym>Synonym)
Description	Extract best answer
Parameters	Content – a hash that type as key, paragraph as value keyword – a list of user query keyword

	Synonym –a list of synonyms of key words
Return Value	Answer

2.3.4 Interaction Diagram

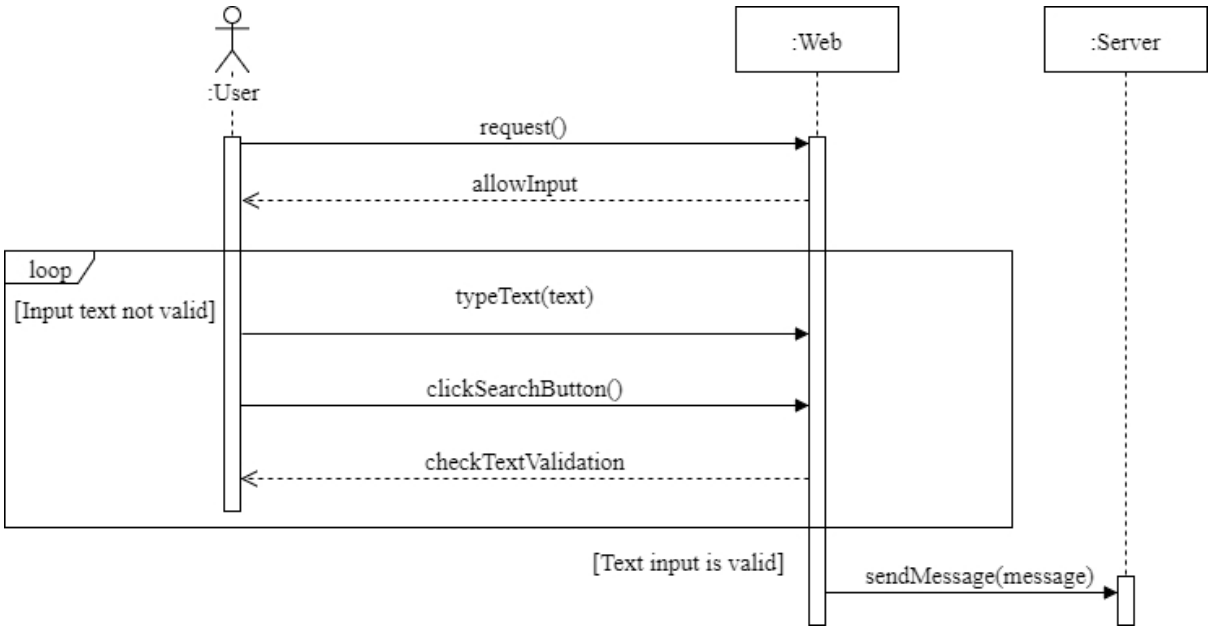


Figure 4: Request Enter Diagram

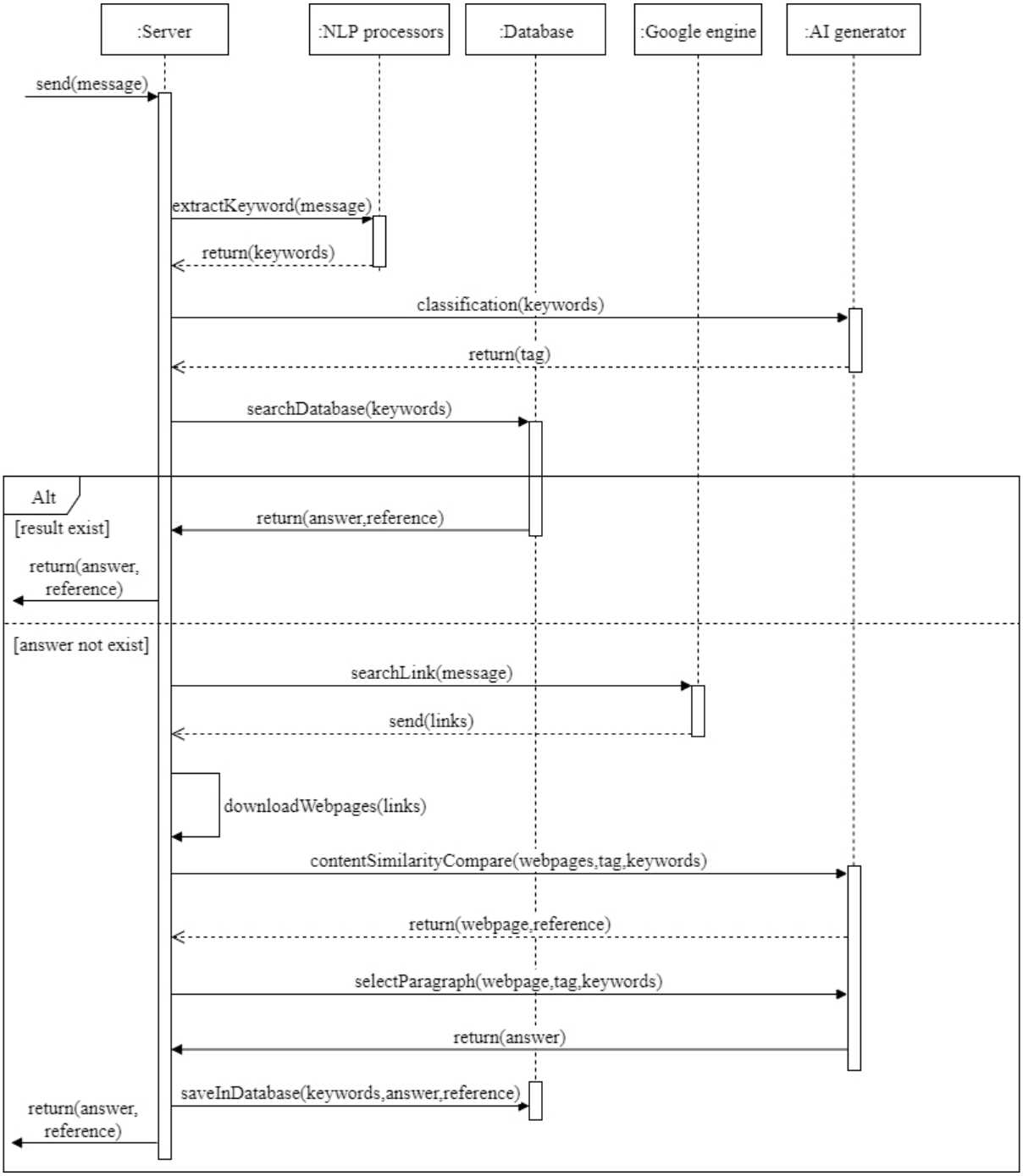


Figure 5: Answer Search Diagram

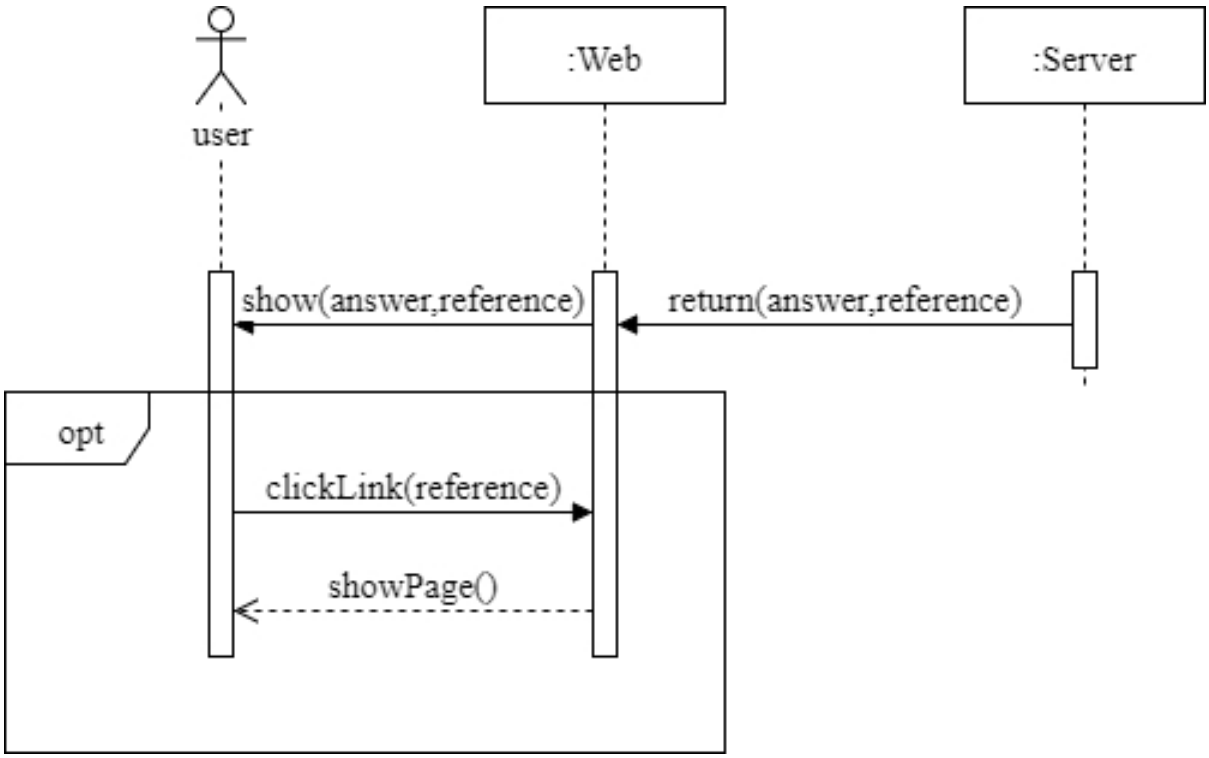


Figure 6: Answer showing diagram

2.4 DATABASE DESIGN

2.4.1 Background

The database is essential for a network application and an underlying framework of the information system of supermarket, bank and library etc. [4]. Database always take parts in the software system as data storage and timely process work. With retrieving data from database, system would always be efficient and according to the process part and data storage part split, data would be safer. Therefore, the Retrieve-Process-Update model is becoming more essential when needing to process a medium or large quantity of data [5].

In our precise search engine system, we apply the database to store the answer of each different search. This is because if several users input the same questions or same meaning of the question, system could process the question and return the answer while also saves the answer with keyword and class tag in the database, thus when the next time, user with same question can retrieve the answer with SQL directly, faster and stable. Because the database uses keyword as key, the system avoids storing reduplicated content. Furthermore, the system sets weight to the keywords for the emphasis of the question and answer from the search count.

The structure of database applied in the system can be illustrated as data dictionary and entity relational model. Each attribute of the table will be assigned some properties to set constraint, including data type and size. These properties are important for the stability of database.

2.4.2 Data Dictionaries

The database for the system stores the data have processed as the answer for the next same question. Also, when system retrieves several existing answer for a question, it would reprocess the keyword more accurately according to the weight. Furthermore, it returns web pages as references.

Entity Description Dictionary

The dictionary table describes entity system, including description, it possible alias called and understand, and their usage in the system.

Entity	Description	Aliases	Occurrence
Keyword	Keyword retrieved from questions	Keyword question	The most important part of a question content.

Attribute Description Dictionary

The table describes every attribute of each entity, including their description, data type and length, if possible to be NULL, and the role played in their entity table.

Entity	Attributes	Description	Data Type and Length	Null	Key
Keyword	WordID	The number and date of reference of keyword	INT,11	No	primary
Keyword	Keyword	Keyword retrieved from questions	VARCHAR,1000	No	NULL
Keyword	Date	Date insert the keyword	Date,8	No	NULL
Keyword	Answer	Answerfor the keyword question	VARCHAR,1000	No	NULL
Keyword	Reference	Reference webpages to answer	VARCHAR,300	No	NULL

Entity-Relation Diagram

The ER Diagram shows the relationship and entity of the precision search system.

Figure 7: Entity relation diagram

Entity

- Log Table:
  *LogID: Integer with 10 length, not NULL, Primary key
  *Date: DATE with 8 length, not NULL.
  *Detal:VARCHAR with 200 length, not null, unique attribute
- Keyword Table:
  *WordID: Integer with 11 length, not NULL, Primary Key
  *Date: DATE with 8 lengths, not NULL
  *Keyword: VARCHAR with maximum 10 length, not NULL, Unique attribute
  *Answer: VARCHAR with maximum 140 length, not NULL
  *classTag: VARCHAR with 10 maximum length,not NULL
  *Count: Integer with maximum 5 length, not NULL
- Answer Table:
  *WebpageID: Integer with 11 length, not NULL, Primary Key
  *Keyword: VARCHAR with maximum 10 length, not NULL
  *Reference: VARCHAR with maximum 100 length, not NULL, unique attributes

Relationship

- Contain:
  Between Keyword table and Answer Table, one key word may extend many answers but one answer is related to just one keyword.

- Record:
  Between Log table and Keyword table. One log has one key word and one keyword to one log.

- Record:
  Between Log table and Answer table. One log has one answer and one answer to one log.

2.5 USER INTERFACE DESIGN

2.5.1 Introduction

The user interface is based on the web platform. The webpage will be used to input and shows results to users. For this webpage, some criteria should be taken into account.

- First, operation should be easy. Since this engine is used by multiple kinds of people, the webpage is designed as traditional search engine type.
- Second, the layout should be clear. Excepting necessary buttons and display block, there is little decoration which might perplex users.

In this case, the draft of the webpage is designed as below:

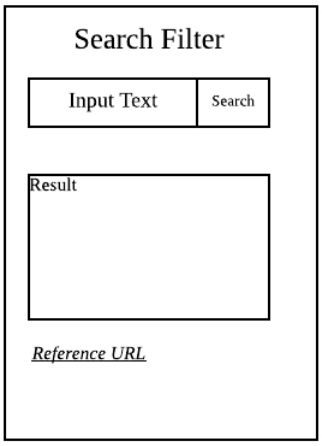


Figure 8: User interface design

To specify the webpage, this design consists of three parts:

1. Webpage elements introduction: describes function and linking of elements
2. User webpage state diagram: indicates the various states of webpages used on different operations
3. States description: introduces the functionality and operation of different states

2.5.2 Element Introduction

Element name	Function
Searching bar	Used for type input Clicking triggers cleaning up and re-input
“Search” icon	Encrypt text from searching bar Link server Send text to the server
“Answer” block	Create when web receive answer from server Display answer with reference
Reference link	Create for further use

2.5.3 User Webpage State Diagram

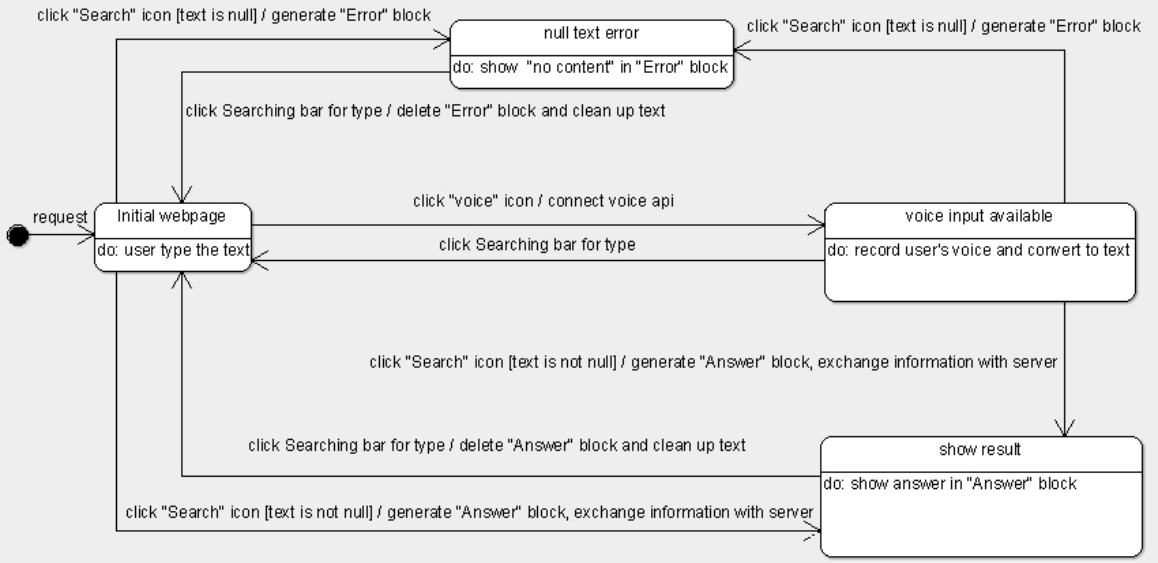


Figure 9: User webpage state diagram

2.5.4 State Description

State name	Elements	Description
Initial state	Searching bar Voice icon “Search” icon	When users request this webpage or user click searching bar, the UI turn to this page. The system is activated and ready for user to type in the searching bar.
Voice input available	Searching bar Voice icon - Voice device open “Search” icon	When users click Voice icon, the voice input API is connected and the input pattern turn to voice input
Show result	Searching bar Voice icon “Search” icon “Answer” block - with “Reference” block	When users click “Search” icon and text is not null. The text will be encrypted and send to server for processing. Then server will return the answer with reference to UI. The “Answer” block is created and display the answer.
Null text error	Searching bar Voice icon “Search” icon “Error” block	When users click “Search” icon and text is null. The “Error” block is created and display “no content” and require users to input again.

2.6 TESTING DESIGN

2.6.1 Background

The testing part is to test if the system made could run satisfied with several constraints then we could say that this software meets user's needs. To ensure system run correctly, we would apply 4 regular testing and 1 specific testing and evaluate the result by different evaluation criteria set before implementing the system. These testing will help us developers more quickly find the defects and evaluate the efficiency of each module to the whole system.

2.6.2 Evaluation

Obviously, the main advantage of our system is that user could get answers quickly.

Furthermore, as is shown on the search page, you could use it as convenient as google search. However, this search filter could not give as many results as Google as the cost of getting a focus and quick answer. On the other hand, because people tend to spend a fast-paced life, getting information quickly has become the main trend, in this regard, our system is more suitable to nowadays' people habits.

2.6.3 Regular Testing

- Structural testing
  Criterion:
  Each module would not exist any structural or syntax errors
  Methods:
  Firstly, write with the help of compiler to find the syntax errors. Then, use up-bottom to split the module and find which module has problems

- Interface testing
  Criterion:
  1. User could recognize where to input the question and find the processed answer.
  2. The parameters and methods are transmitted correctly between different modules

  Methods:
  1. Design the web page with placeholder in the input area like 'Please input your question here', 'The answer is...'
  2. Write something like java interface to define the name and parameters of the methods and ask all team members to follow them.

- Integration testing
  Criterion:
  The system could give us the answer we want around 140 words when given an objective question like: "When was America established?"
  Methods:
  Choose around 10 questions and then input them one by one and check if it can retrieve relevant topic, then if the sentences are fluent.

- Stress testing
  Criterion:

1. The system should at least undertake 200 users to use concurrently.
2. The system input should detect and address illegal input.

Methods:

1 . A testing program could be encoded that it could output 200 different question with 200 threads. Then we could test if it could receive the relative answer.

2 . Input something like script or SQL insert to test if the system could process normally.

2.6.4 Specific Testing

- Algorithm

Criterion:

1. The algorithm should have less coupling between each other.
2. The combine algorithm should give us a fluent sentence.
3. The algorithm should have some accuracy and efficiency.

Methods:

1. Check the use of parameters and functions if they are linked to other algorithms after one algorithm was implemented.
2. Randomly input some objective questions includes extreme one and check if the content in command line is fluent and topic-related.
3. Test the space use and time use in a single question and then increase with the length and quantity of questions and record the time and space.

3 PROJECT PLANNING

3.1 PROGRESS DESCTIPTION

3.1.1 Original Plan

In the requirement phase, the original plan for the design phase is to divide it into the logical design section and physical design section. In the logical design, system introduction, algorithm design and testing design are expected. Meanwhile, server design, database design and user interface design are included.

- In the system introduction, a functional description and a use case will be shown.
- For algorithm design which is the core part in the design phase, pseudo code will be provided.
- In the server design part, two diagrams that are class diagram and interaction diagram will be drawn.
- For database design, a data dictionary and an entity-relationship diagram are contained.
- A website diagram will be offered for user interface design.
- In the testing design, the testing methods mentioned in the requirement phase will be demonstrated in detail and the whole system will be evaluated.

3.1.2 Progress Review

Up to now, the tasks mentioned above are completed.

- System introduction was finished by Zeyu Liao where use case (see 2.1) is provided.
- Algorithm design was mainly completed by Yan Mo and the pseudo code (see 2.2) was also written with the help of Zeyu Liao.
- Server design was accomplished by Zeyu Liao who is responsible for class diagram and Haomin Song who takes charge of the interaction chart.
- Database design was finished by Tianhui Zhang where a data dictionary (see 2.4) and an entity-relationship diagram (see figure 7) were offered.
- User interface was completed by Haomin Song.
- Gantt chart which can be checked in the appendix and planning change description was written by Kun Wang.

In the next implementation stage, the progress will follow the Gantt chart. All the tasks and member assignment might be modified in the later phase according to the reality situation. After the designing phase, every member of the team will start preparing to implement the project.

The Gantt chart is put in the appendix.

3.2 PLANNING CHANGES

- In the database part, the classification tag is added in order to help search for the exiting problem.
- For the whole system, we add a search limitation instead of allowing all kinds of queries to be entered. The limitations are: who, when, what, how&why.
- We plan to allocate particular expression algorithm to each classification tag.
- In the interface part, the design principle which means why we design like this is added.
- In the system description part, we add a clear statement about what API we will use: voice input API and keyword crawlwebpage to extract keywords in the query.

4 REFERENCE

[1] *The Advantages of Using Search Engines*, Online Available:

[2] Kim Y, Convolutional Neural Networks for Sentence Classification, 2014

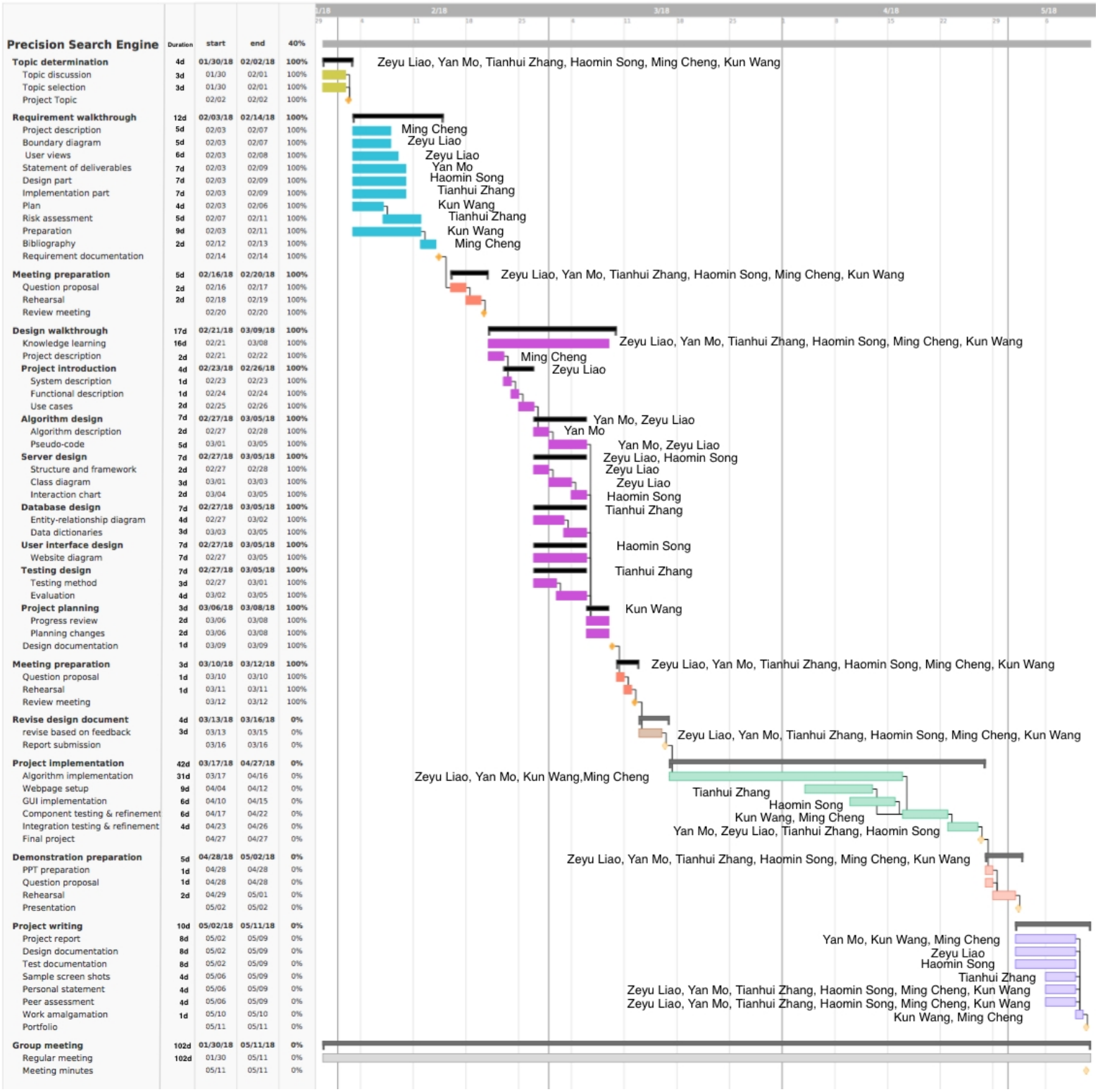
[3] (2015, January 12 Britz D, *WILDML,* Online Available:

http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow

[4] Connolly, T.M. and Begg, C.E., 2005. Database systems: a practical approach to design, implementation, and management. Pearson Education.

[5] Chin F Y, Ozsoyoglu G. Statistical database design[J]. ACM Transactions on Database Systems, 1981, 6(1):113-139.

5 APPENDIX I: GANTT CHART



COMP 214 Test Report

COMP 214 Group 6

Content

1. INTRODUCTION

The Software Test Plan designs to prescribe the scope, approach, resources of all testing activities. The plan must identify the items to be tested, the features to be tested and the types of testing to be performed.

2. TEST DESIGN

2.1Testing Strategy

- Define testing purpose
- Test every module of the software
- Test desired features
- Define Pass/Fail criteria

2.2 Program Modules

In our system, we have program modules containing:

- Front-End Webpage
- Database Retrieve
- Keyword Extraction
- Crawl Snippet
- Best Link Generation
- Best Paragraph Generation

2.3 Features to be tested

- System can currently generate a URL related to user Query
- System can select the most appropriate paragraph from the selected URL

2.4 Approach

2.4.1 Component Testing

For the component Testing, a test script that invoke all component has been used to iterate with different input value as well as generate testing log.

2.4.2 Database Testing

We test database mainly for testing the retrieve speed, accuracy of our query result and the most important, what if we have a break during the SQL operation.

2.4.3Whole system testing

For the whole system testing, we have designed time testing integration testing and Stress testing.

2.4.4 Performance Testing

Due to our system only have one user interaction interface, it is our first assignment to test the reaction time from the user submits the question to the time that webpage returns the answer and reference. In this section, we divide it into 3 parts: the answer already in the database, answer in Google feature snippet and answer that retrieve from the best page and paragraph, and then take “DOM Content loaded” as the reaction time to compare. We are here applying Nah’s data that people are willing to waiting time as consult [1]. Considering our server not as excellent as big companies’, we set the acceptable time to 5 seconds.

2.4.5 Stress Testing

For stress testing, in the real testing, we design 6-group members to search different questions at the same time. Then we use Page speed tools provided by Google to test when 100 users use our web page.

2.4.6 Integration testing

When it comes to integration testing, we prepare 20 questions with three different kinds. In addition, we test each question for

1. Whether there is a valid answer.
2. Whether the answer could match answer?
3. Whether the reference has a great relation with the question?

2.5 Pass / Fail Criteria

2.5.1 Suspension Criteria

Database:

- query time larger than 1s or insert or update failed but not return to last state

System:

- Return no answers
- Answer and Reference cannot match question
- Less than 100 users use our system
- Load time more than 5 seconds

Keyword Extraction:

- Keyword has marked as noise
- No keyword returned

Snippet:

- Return null while there exists snippet on Google
- Snippet cost more than 2 second to return

Best Link:

- No link returns
- Irrelative link return

Best Paragraph:

- No paragraph returns
- Irrelative paragraph return

2.5.2 Resumption Criteria

Database: Query time larger than 0. 5 second, smaller than 1 second

System: Answer or Reference only one cannot match question

Keyword Extraction: All keyword Returned

Snippet: Snippet correctly return within 0.5 – 1 second

Best Link: link relative to user query return

Best Paragraph: Paragraph related to user query return

2.5.3 Approval Criteria:

Database: Query time smaller than 0.1 second and insert or update successes

System: Answer and Reference both match question & Load time less than 5 seconds

Keyword Extraction: Return all keyword and the data noise has been minimize

Snippet: Snippet correctly return within 0.5 second

Best Link: Link relative to user query return

Best Paragraph: Paragraph highly related to user query return

2.6 Testing Process

2.5.2 Testing Tasks

- Component Test
- Integration Test

2.7 Environmental Requirements

2.5.3 Hardware

- Debian Stretch
- Core 1 virtual Intel Xeon E5-26
- 1 GB virtual RAM

2.5.4 Tools

- MySQL
- Django
- Google Page Speed
- Load Impact

3. TEST PART

3.1 Test Components

3.1.1 Keyword Extraction

In the Keyword Extract Testing, module KewordExtract.py will be tested to see if it generated a list of sorted keywords according to the importance. A set of user queries will be passed to the module and the following table will show the result of Keyword Extract testing.

User Query	Criteria	Expecte Output	Actual Output	Status
The Light Phone	Extract Userful Inofomation as list, and sort according to importance	[‘Phone’, ‘Light’]	[‘Phone’, ‘Light’]	Pass
Junit Testing Principle	Extract Userful Inofomation as list, and sort according to importance	[‘Junit’, ‘Testing’, ‘Principle’]	[‘Junit’, ‘Testing’, ‘Principle’]	pass
Michele Zito University of Liverpool	Extract Userful Inofomation as list, and sort according to importance	[‘Michele’, ‘Zito’, ‘University’, ‘Liverpool’]	[‘Unviersity’, ‘Zito’, ‘Liverpool’, ‘Michele’]	Pass

Dr Danushka Bolleala Profile	Extract Userful Inofomation as list, and sort according to importance	[‘Danushka’, ‘Bollegala’, ‘Dr’, ‘Profile’]	[‘Danushka’, ‘Bollegala’, ‘Dr’, ‘Profile’]	Pass

3.1.2 Snippet Crawl

In the Snippet Crawl Testing, module download_snippets.py will be tested to see if successfully crawl google featured snippets with user Query.

User Query	Criteria	Expected Output	Actual Output	Status
Who is Trump	Return Snippets if exist one	Donald Trump 45 U.S President	Donald Trump45th U.S. PresidentDonald John Trump is the 45th and current President of the United States, in office since January 20, 2017 …….	pass
The Light Phone	Return Snippets if exist one	There is no Snippet for this Query	There is no Snippet for this Query	pass

3.1.3 Best Link Generation

In this Best Link Generation Test, module bestLink.py will be tested to see if it generates an appropriate link for a list of google link results according to the user query.

User Query	Criteria	Expected Output	Actual Output	Status
The light Phone	Generate an appropriate URL with relative Title	Title: The Light Phone	Title: The Light Phone – The Hustle	pass

Junit Testing Principle	Generate an appropriate URL with relative Title	Title: Junit Test Principle	Title: Java Unit Testing Principles with Junit	pass
Dr Danushka Bollegala Profile	Generate an appropriate URL with relative Title	Title: Danushka Bollegala	Title: Machine Learning on Mobile – Source Diving	pass
Michele Zito University of Liverpool	Generate an appropriate URL with relative Title	Title: Michele Zito	Title: Michele Zito | Facebook	pass

3.1.4 Best Paragraph Generation

In Best Paragraph Generation Test, module best_answer.py will be tested to see if it generates a reasonable snippet from the content of the best link according to the user query.

User Query	Criteria	Expected Output	Actual Output	Status
The light Phone	Generate a paragraph that related to Query	The light phone is ...	The Light Phone is anti-technology technology. Built to keep you connected, but only just enough...	pass
Junit Testing Principle	Generate a paragraph that related to Query	Junit Testing is ...	JUnit is an open source unit-testing framework for the Java programming language...	pass
Dr Danushka Bollegala Profile	Generate a paragraph that related to Query	Dr Danushka Bollegala is ….	Recently, Dr Danushka Bollegala who has been researching NLP, machine learning and data mining ….	pass
Michele Zito	Generate a	Michele Zito is a...	Michele Zito's profile	pass

University of Liverpool	paragraph that related to Query		photo, Image may contain: 1 person Michele Zito is on Facebook...	

3.1.5 Database Testing

After testing, we found that every SQL sentence could be run in 0.01 second as we always fetch one or two items because every time we could receive message that "Query in 0.0s". Furthermore, as we use commit and close the connection after each query. Our database part runs as expected. In conclusion, database passes the test.

3.2 Test System

3.2.1 Performance testing:

Firstly, on condition that some users else have already searched the same question. Then system could read the answer from the database directly.

Table 1 Reaction Time of web page

	Answer in Database	Answer in Feature Snippet	Answer to calculate for best page and paragraph
1 Test case	375 ms	861ms	4.00s
5 Test case	402ms	847ms	4.18s
20 Test case	443ms	943ms	4.97s

As is shown in the table, DOMContent can be loaded completely in 500 ms in average and people could wait up to 2 second which could satisfy the user's tolerance [1].

Furthermore, if we do not have this question, we put this question into our algorithm to calculate. After the time testing, we found that it is different due to if there is a

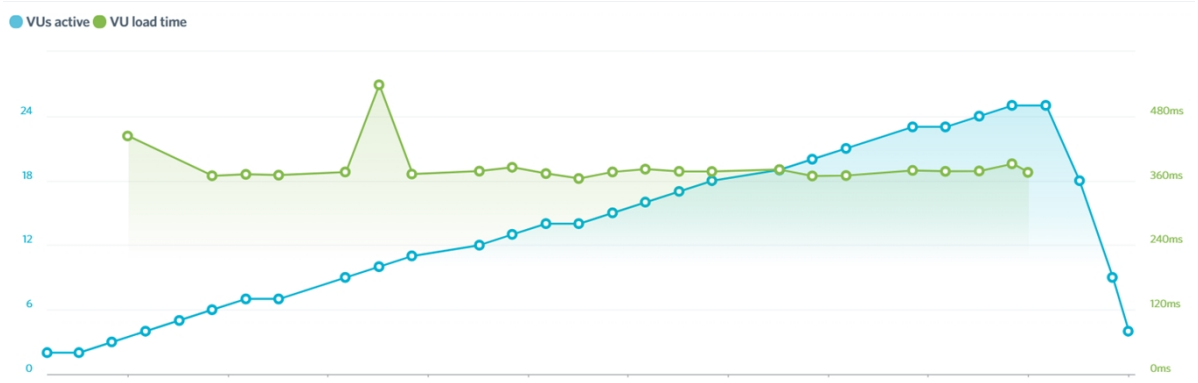
feature snippet to use. If so, the search time could reduce to 1s to 1.5s. Otherwise, it delays about 5 second according to the table.

During the test, we found a situation that when we search for a same question in database, the load reaction time would increase slowly from 300ms to 500ms, this may because the cache of explorer accumulates.

3.2.2 Stress Testing

During our six-member test different questions at the same time, our system works stable and no extra loaded time. When under simulating 100 users surf our web pages. It also works well because there are not much content to load when surfing our web page.

However, when it comes to search in multi-threaded state, loaded time gets an evident increase. In our test, when there are 25 users using search function, the loaded time would increase larger than 480ms as the picture below.



It may mean we have to expand the quality of our server for more users to use. Actually, if we want to make the server that could address 100 people's request, in our assumption, we need at least 2 CPUs and 8G memories for a lower load time.

3.2.3 Integration testing

For the question having featured snippet answer or already stored in the database, the system could always return accurate answer and reference. For example, when input "Who is Donald Trump?" It would return Google's feature snippet. However, it has

some problem with the given answer of the question does not have the featured snippet though the reference of our system could match the question, sometimes the returned result would be given as the previous question's answer. For example, we test a question of "Michele Zito - University of Liverpool", in the server terminal, it could give the correct answer of Michele Zito's Facebook page. However, for the web system it may give the answer of one of last several questions' answer we searched. After testing, we found this situation could happen as a rate of 30%. It may be due to the cache inside python. In addition, we have not found any better solution except readdress the answer, which would waste a lot of time on each question. Therefore, we decided not to use this method for not delay those questions correct.

4. CONCLUSION

After the test, we could conclude that even though all the components could run normally and the web page could return the reference and answer in an acceptable time, we still have some problems, which need a future solution. Firstly, the answer sometimes shows again for another question but reference correct. Secondly, we cannot address too much request for search difficult questions, which increase the need of calculation, or it would increase load time.

5. BIBLIOGRAPHY

[1] F. Nah, "*A study on tolerable waiting time: how long are Web users willing to wait?*" Behaviour & Information Technology, vol. 23, no. 3, pp. 153-163, 2004.

Screen shot

1. Using example screen shot

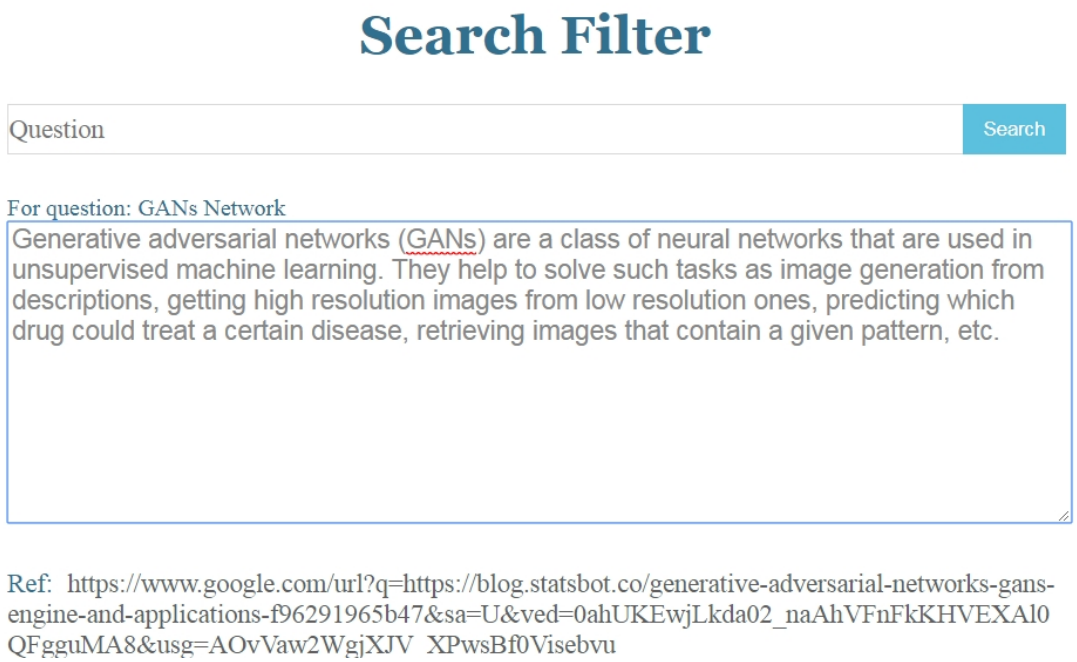


Image 1: Website using example 1

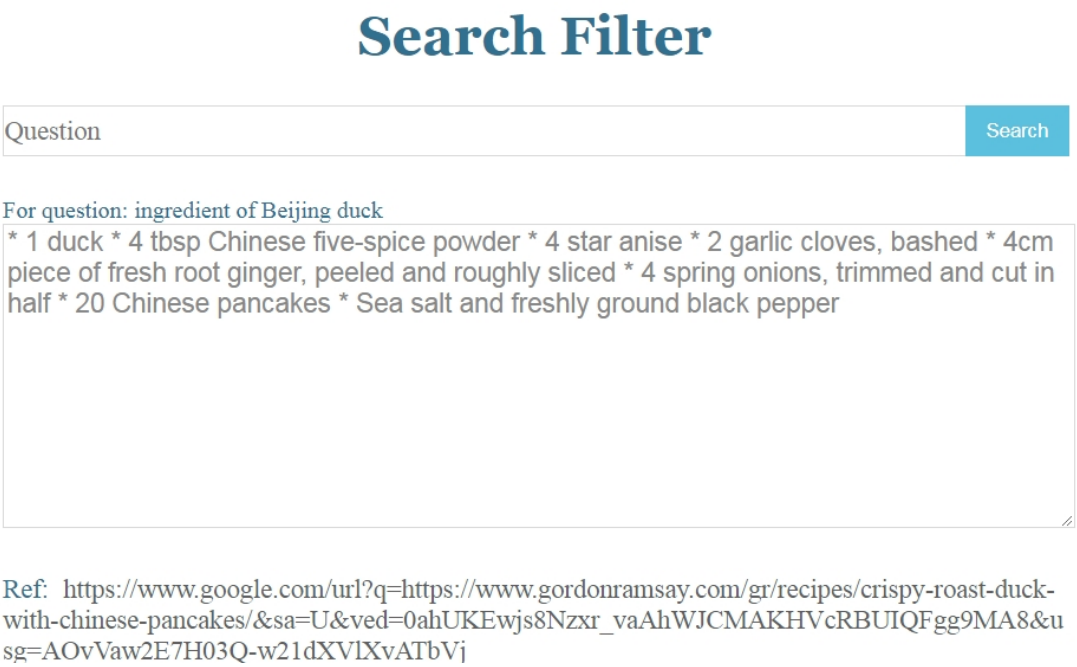


Image 2: Website using example 2

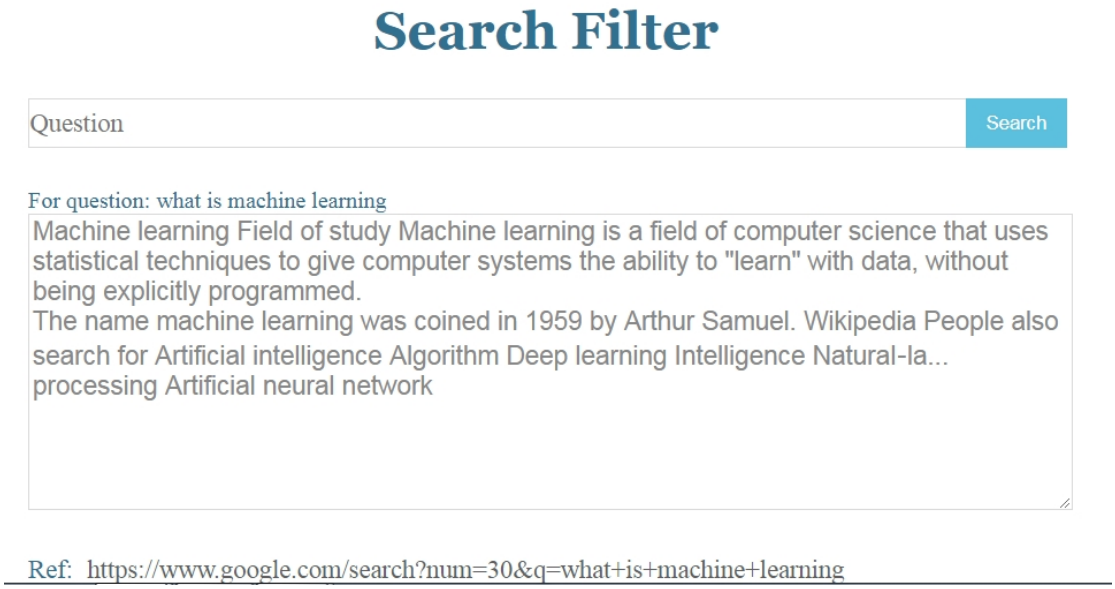


Image 3: Website using example 3

2. Server running screen shot:

Image 4: Test for best paragraph

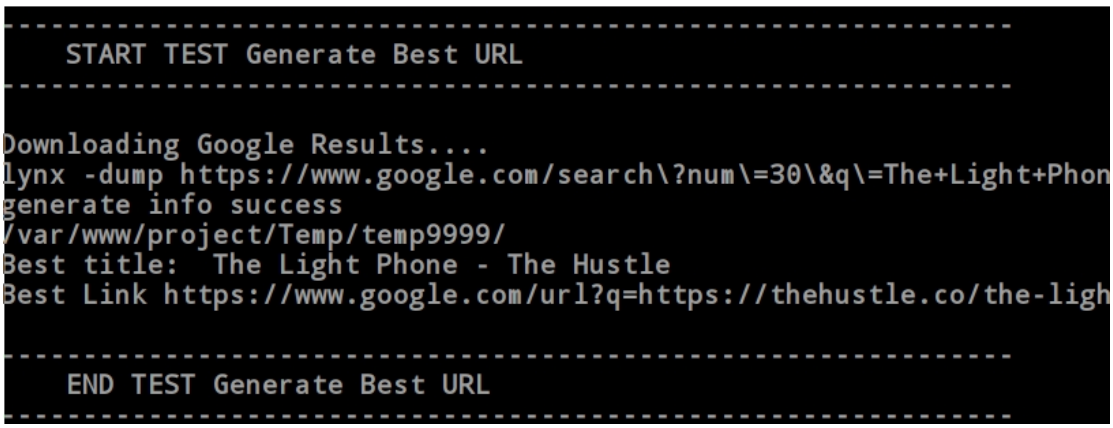


Image 5: Test for best URL

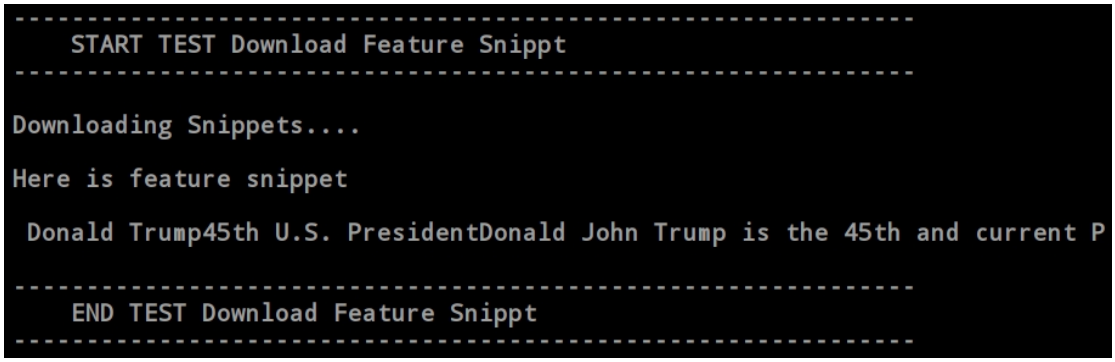


Image 6: Test for featured snippet

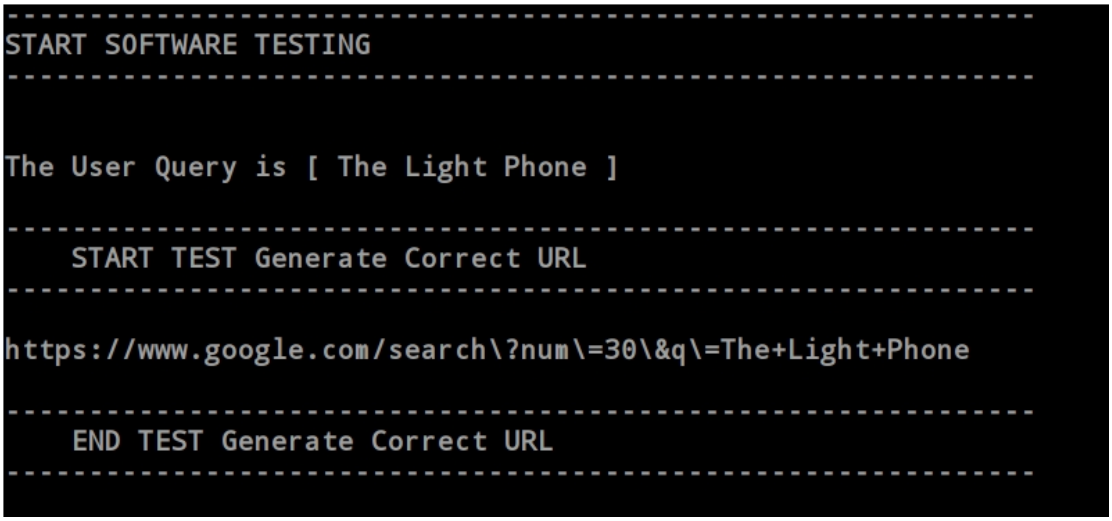


Image 7: Test for Google URL

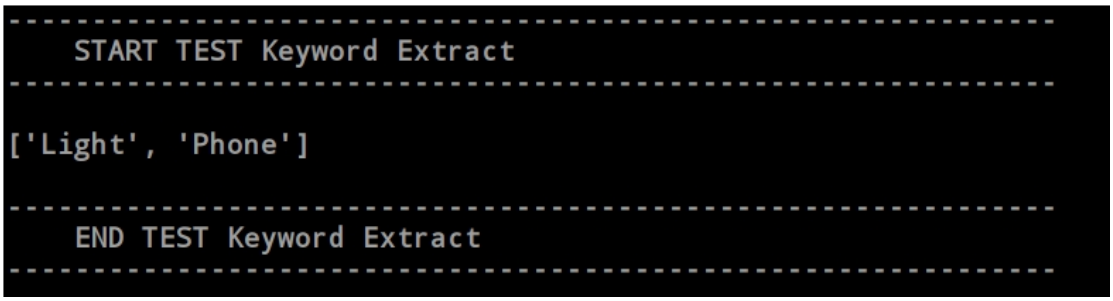


Image 8: Test for keyword extraction