Search Engine for Stevens Academic Courses

Abhilash Ugaonkar
Computer Science Department
Stevens Institute of Technology
Hoboken, USA
augaonka@stevens.edu

Viraj Gite
Computer Science Department
Stevens Institute of Technology
Hoboken, USA
vgite@stevens.edu

***Abstract*— Implementation of Boolean and Vector space retrieval models to build a search engine for course syllabuses. The search engine will help and allow the Stevens Community to search for Courses based on their interests during the course registrations.**

Keywords—Boolean Retrieval Model, Vector Space Model, Inverted Index, Search Engine.

I. INTRODUCTION

Goal – To build a Syllabus Search Engine for Stevens Community

Motivation – Our motivation behind the project topic was to identify real world problem and then apply our knowledge, learning and experience to solve that problem.

1. *Enable Stevens Community to Retrieve Course Details*

Whenever information seeker such as student, faculty wants to know about course and they try to search for the same on the Stevens website they face below problem:

- What is the actual course code?
- What is the actual name of course?
- What is being offered in the course?

They always try to search with some keywords to find the courses they want, and are presented with some or no results. Since user is unaware of these details he ends up facing following problems:

- Go through a turmoil of documents
- Waste of efforts and time
- Non-relevant information that is NO SUCCESS

2. *Understand the behind the Scene Functionality and Complexity of Search Engine*

This was our secondary goal to apply learnings of the course and gain practical knowledge and experience of how things works on large scale.

II. SYSTEM FRAMEWORK

The overall system is distributed into three main modules which are:

- Stevens Intranet and Web Crawler
- Corpus
- Search Engine

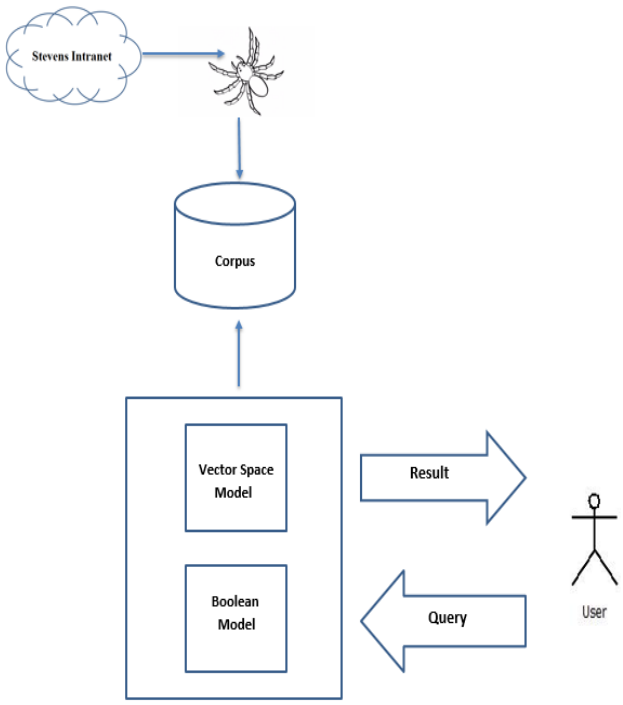

Fig 1: System Framework

The system flow and module functionality can be given as follows:

- Web Crawler crawls the syllabus links on the intranet and converts the syllabus xml files into text files.
- All these text files form a document corpus which is then used by search engine to give response to user query.
- Search engine module is sub-divided into two parts which are Vector Space module and Boolean module which implement the main algorithm of the search.

III. ASSIGNMENT OF TASKS

A. Tasks Assigned to Abhilash

- Implementation of Vector Space Model
- Implementation of Precision and Recall Evaluation Model

Hours: 4 Hours

B. Tasks Assigned to Viraj

- Implementation of Boolean Retrieval Model
- Writing the program for the web crawler

Hours: 4 Hours

IV. DETAILED DESCRIPTION

A. Web Crawler Module

The web crawler module is named the Syllabus downloader.

Function:

To fetch and preprocess XML data available at the given links and store the data in a text file for each link.

Implementation

Data Structures Used:

No data structures are used because the data is read, processed and stored dynamically to a file. Storage of data in the main memory is not required

Open source packages: **[2]**

- ***pyCurl*** – To fetch objects addressed by an URL (Used to fetch syllabus data from the Stevens CS department Intranet).
- ***cElementTree*** – To traverse and process XML tree (The syllabus data available on the Stevens Intranet is of the form XML. The cElementTree package is used to extract the elements Course Name and Topics Covered)

Main Challenges:

The data source for the course syllabuses were PDF files at first. Fetching data from these encoded PDF files was difficult and the module missed most of the required text or sometimes was simply unable to read the file. We had to find a consistent data source that could be parsed and preprocessed to be searched. XML Data available on the Stevens Intranet helped to solve the problem.

B. Boolean Retrieval Module

This module implements the Boolean Retrieval Model of the search engine. **[1]**

Function:

To create an inverted index for all the terms in the syllabuses and search it for the results based on the user given query.

Implementation

Data Structures Used:

The code of this module makes use of two main data structures; **list and dictionary**. **[3]**

The syllabus data read from the files is preprocessed and the final tokenized words are stored in the **list data structure**.
The **dictionary data structure** in python is used to create an inverted index, where the key is the term/word and the value is a set of documents that contain that term. The value field in the dictionary data structure is modified to allow setting multiple values for the same key.

Open source packages: **[2]**

- ***Nltk – modules***
  - RegexpTokenizer – used for tokenizing.
  - get_stop_words – used to remove stopwords from the syllabus.
  - PorterStemmer – used to stem long words.
- ***Collections –*** The program uses the dictionary data structure from the collections package in python.

Main Challenges:

Implementing the inverted index and writing the programming logic for all possible query inputs was one of the main challenges in implementing this module. Dictionary data structure in python helped to eliminate this problem by tweaking the value field into a set.

C. Vector Space Information Retrieval Module

This module implements the Vector Space Model of the search engine. **[1]**

Function

To calculate an inverted index of every term in the syllabus file. It asks you to enter a search query, and then returns all documents matching the query, in decreasing order of **cosine similarity.**

Implementation

Data Structures Used:

The implementation of this module makes use of two data structures: **set, list and dictionary**. **[3]**

The syllabus data read from the files is preprocessed and the final tokenized words are stored in the **dictionary**.
The **dictionary data structure** in python is used to create an inverted index, where the key is the term/word and the value is a set of documents that contain that term. The value field in the dictionary data structure is modified to allow setting multiple values for the same key.

Open source packages: **[2]**

- ***Nltk – modules***
  - RegexpTokenizer – used for tokenizing.
  - get_stop_words – used to remove stopwords from the syllabus.
  - PorterStemmer – used to stem long words.

- ***Collections –*** The program uses the dictionary data structure from the collections package in python.

Main Challenges:

Implementing dictionary of the input syllabus file terms and value followed by matching the user query set with the dictionary was most challenging part. Python set operations such as union, intersection, sorted made query matching and cosine similarity score calculations a lot easier.

D. Precision and Recall Evaluation Module

Function

To evaluate the query against corpus and determine precision and recall values of the Boolean Model and Vector Space Model against the similar queries.

Implementation

Data Structures Used:
The implementation of this module makes use of two data structures: **set and dictionary**. **[3]**

The **dictionary data structure** used in this module contained user query as key and relevant documents as values. Relevant documents are defined with golden standards. User entered query is stored in the set and matched against keys in the dictionary. Once the key is matched then retrieved documents are matched against the values (relevant golden standard documents).

Gold Standards: **[1]**

Relevant documents are defined with golden standards. Gold standard were assigned as by below:

i. Start with the corpus documents
ii. Create a set of queries for the corpus
iii. Label relevant document for each query

Open source packages: **[1]**

- **Pretty Table –** This is package used for displaying tabular data in a visually appealing ASCII table format. This is used to display the result in tabular format.
- **RE –** This package provides regular expression matching operations. It used in query evaluation.

Main Challenges:

The main challenge we faced was to select the relevant documents as per the gold standards. Creating golden set of documents for different queries was very time consuming.

V. EVALUATION

Dataset

We formed our corpus by crawling intranet URL of Stevens Institute of Technology **[4]**. The dataset consists of overall forty-six files.

Precision and Recall

We calculated Precision and Recall **[1]** values to evaluate the performance of Boolean and Vector Space Models.

Precision: It can be defined as the fraction of retrieved documents that are relevant = **P (relevant | retrieved)**

Recall: It can be defined as the fraction of relevant documents that are retrieved = **P (retrieved | relevant)**.

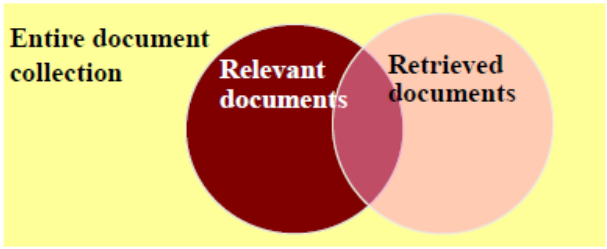

Fig 2: Retrieval Evaluation

We assigned determined the Precision and Recall for below two queries:

Query 1:
*Data **and** Courses **or** Mining* ….. (For Boolean Model)
Data Courses Mining ….. (For Vector Space Model)

Relevant Documents as per Gold Standard for above query were: {1, 4, 6, 7, 12, 13, 15, 21, 22, 39, 41, 42}
Results:

Boolean Model Retrieved documents:
{1, 4, 6, 7, 12, 13, 15, 21, 22, 39, 41, 42}

Precision and Recall = 1.0

Vector Space Model Retrieved Documents:
{41, 18, 21, 6, 40, 35, 23, 30, 13, 4, 43, 14, 37, 1, 36, 10, 22, 7, 38, 11, 31, 28, 8}

Precision = 0.3636 and Recall= 0.6666

Query 2:
*CS **and** 609* ….(For Boolean Model)
CS 609 ….(For Vector Space Model)

Relevant Documents as per Gold Standard for above query were: {46}
Results:

Boolean Model Retrieved documents: {46}

Precision and Recall = 1

Vector Space Model Retrieved:
{46,1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31 ,32 ,33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45 }

Precision = 0.26 and Recall = 1

Since Boolean Model uses logical operators they have higher precision and recall values however Vector Space Model make the partial matching and hence retrieves the partial matching documents as well.
Since Boolean Model uses logical operators they have higher precision and recall values however Vector Space Model make the partial matching and hence retrieves the partial matching documents as well.

VI. Conclusion

After successfully implementing the Boolean and Vector Space Model we conclude that Boolean Model is easy to implement however logical operators AND, OR and NOT makes it too rigid. It results in too few or too many results. Also, the retrieval of documents is based on the binary operation which doesn't reflect the human intuitions. On the other hand, Vector Space model is complex to implement but is not rigid as Boolean model and retrieves the partial matches. It retrieves the documents in the decreasing order of cosine similarity which makes it easier for information seeker to get most relevant documents at first. Vector Space model can be easily scaled for large collection of documents.

As future scope for this project there is lot of scope to improve these existing models. We can include the Natural Language support which can help us address the weaknesses of the Vector Space model. Also, we can add the semantic modules, synonyms and homonyms for more sophisticated information retrieval. We can also improve code efficiency by implementing Trie data structure.

References

[1] Prof. Wendy Wang: Lecture Slides
[2] https://pypi.python.org/pypi: Python Packages
[3] https://docs.python.org/3/tutorial/datastructures.html: Python Data Structures
[4] https://my.stevens.edu/ses/cs/graduate/courses: Stevens Intranet