**INFORMATION RETRIEVAL API BASED ON NLP TECHNIQUES**

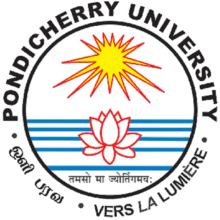
By

**Sourabh Roy**

(Registration Number : 17352227)

**Project report submitted in partial fulfilment of the requirements for the award of the degree of**

**MASTER OF COMPUTER APPLICATIONS**



**DEPARTMENT OF COMPUTER SCIENCE**

**SCHOOL OF ENGINEERING AND TECHNOLOGY**

**PONDICHERRY UNIVERSITY, KARAIKAL CAMPUS**

**June 2020**

**BONAFIDE CERTIFICATE**

This is to certify that this project work entitled “**INFORMATION RETRIEVAL API BASED ON NLP TECHNIQUES**” is a bonafide record of work done by **Mr. Sourabh Roy**, in partial fulfilment for the degree of Master of Computer Application at Pondicherry University.

This work has not been submitted elsewhere for the award of any other degree to the best of our knowledge.

**INTERNAL GUIDE HEAD OF THE DEPARTMENT**

Mr. G. Amirthyogam Guest Faculty **Dr. G. Kumaravelan**

Department of Computer Science Assistant Professor and Head(i/c)

School of engineering and technology Department of Computer Science

Pondicherry University, Karaikal Campus School of engineering and technology

Karaikal – 609 605 Pondicherry University, Karaikal Campus - 609 605

Submitted for the Viva-Voce Examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

**TABLE OF CONTENTS**

**Title Page No**

**ACKNOWLEDGEMENT 1**

**SYNOPSIS 2**

1. **INTRODUCTION 3**
2. **PROBLEM DEFINATION AND FEASIBILITY ANALYSIS 8**
3. **SOFTWARE REQUIREMENTS SPECIFICATION 10**
4. **DEVELOPMENT LIFE CYCLE 13**
5. **SYSTEM ANALYSIS 14**
6. **SYSTEM DESIGN 15**
7. **IMPLEMENTATION AND TESTING 18**
8. **CONCLUSION AND FORESEEABLE ENHANCEMENTS 20**

**BIBLIOGRAPHY 21**

**APPENDIX 22**

**ACKNOWLEDGEMENT**

In the accomplishment of this project successfully, many people have best owned upon me their blessings and the heart pledged support, this time I am utilizing to thank all the people who have been concerned with this project.

Primarily I would thank god, for being able to complete this project with success. Then I would like to thank my Internal guide Mr. G. Amirthayogam to stand with my ideas and ensuring full support. I thank my Head of the Dept. Dr. G. Kumaravelan for providing us amenities and opportunities.

Last I would like to thank Mr. Mikhail Mirzayanov, faculty at HSU who have helped with this project a lot by providing platform to develop my skills and motivating me.

Although, the report has been prepared with utmost care and deep rooted interest. Even then I accept respondent and imperfection.

**SYNOPSIS**

The meaning of the term information retrieval can be very broad. Just getting a credit card out of your wallet so that you can type in the card number is a form of information retrieval. However, as an academic field of study, Information retrieval might be defined thus :

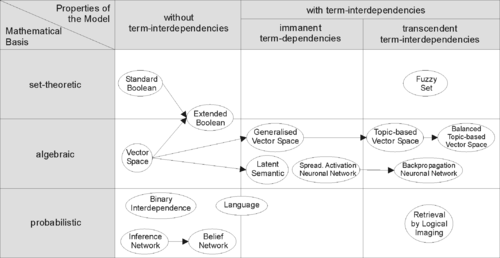
Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

As defined in this way, information retrieval used to be an activity that only a few people engaged in: reference librarians, paralegals, and similar professional searchers. Now the world has changed, and hundreds of millions of people engage in information retrieval every day when they use a web search engine or search their email. Information retrieval is fast becoming the dominant form of information access, overtaking traditional database style searching (the sort that is going on when a clerk says to you: “I’m sorry, I can only look up your order if you can give me your Order ID”). IR can also cover other kinds of data and information problems beyond that specified in the core definition above. The term “unstructured data” refers to data which does not have clear, semantically overt, easy-for-a-computer structure. It is the opposite of structured data, the canonical example of which is a relational database, of the sort companies usually use to maintain product inventories and personnel records. In reality, almost no data are truly “unstructured”. This is definitely true of all text data if you count the latent linguistic structure of human languages and as mentioned before, this project is a work on developing a API library that can be used by users ( by users I am pertaining to developers and hobbyists ) to develop applications, practice and learn without the need to explore into deeper concepts and implementation details. This project deals with the greater domain of topics like Programming, Algorithms, Data structure, Software engineering, Natural Language Processing, Template metaprogramming and multithreading. This domains greatly add flavours to success of this project, therefore creating a rich, efficient yet lucid API that can be used with fun. Many programmers when trying to work with problems in NLP face language barrier on their way as mostly python has been chosen by data scientists and engineer to solve problems n NLP mainly because of easy to learn and trace bugs. Now worldwide there are more than 8.2 million python developers contributing their skills and thus developing a monopoly with python over other efficient languages. Recently IOT is in buzz and there is a increase in demand of skills in Artificial Intelligence and efficient programming language to solve and build libraries problems in various sub domains in artificial intelligence like NLP just because of the raw power exhibited by efficient languages. There are various libraries out in the market that deals with Information Retrieval problems that are computationally touched its peak capability but those libraries lack in speed and size. By speed I have meant processing speed, as we know NLP and ML problems require to use high density data and to process this humongous data require exceptional CPU clock speed. By size I have meant storage space required to successfully provide result of an operation, as any problem we pick from NLP requires huge data to process. Keeping this as forefront, I have decided to solve this in my project.

**INTRODUCTION**

*Information Retrieval* ( IR ) is a system defined as a program that deals with the organization, storage, retrieval and evaluation of information from document repositories. It is heavily dependent on textual information.

Before Pre – IR era, computer scientists had tough time to validate the accuracy of two or more documents with each other. This led to influence mathematical theories to work around this problem. Later advanced GPU helped a greater leap of *Artificial Intelligence* theories and various models have proven to solve these problems.



Basically there are of threeda types:

1. **Classical IR Model** : It is modelled based on mathematical knowledge that is easily recognized and understood.

* *Boolean*
* *statistic*
* *Vector*
* *Probabilistic*

1. **Non Classical IR model :** It is total opposite of classical IR model where logic model, situation theory model and interaction models are used.
2. **Alternative IR model** : Actually it is a enhancement of classical model where some kind of transformation are used and fed on classical IR model. Also recurrent neural network and fuzzy logic are used to develop this kind of hybrid models.

In this project classical and alternative IR model theory has been used and has been discussed further.

I have used some of the techniques from classical and alternative IR model because :

1. Easy to implement.
2. Independent of other statistical and machine learning API’s.
3. Effective on most problems related to IR NLP.
4. Less time to develop.

Each techniques I have used here have unique properties and can be used on varied problems, some of them are defined below :

Various types of information retrieval models :

1. **Jaccard similarity** :

* It is also known as intersection over union and it is used to gauge similarity and diversity between finite set of words.
* It is particularly helpful for duplicate detection.
* To find this, we have to find J(doc1,doc2) = (doc1 ∩ doc2) / (doc1 ∪ doc2)
* Jaccard similarity will not change when same word is repeated more than once whereas in cosine similarity the value will be affected.
* Jaccard similarity can be particularly useful for duplicates detection.

**Advantages**

* Good for identifying mirror sites.
* Best works for search based results.

1. **Cosine similarity** :

* Cosine similarity : (Best for finding dissimilarity between documents based on frequency of words in a document)
* Classical approach to find content overlap between documents.
* Finding arc cos angle between two documents.
* Here documents are denoted as vectors and Words represent dimension of vectors.
* We find occurences of words in a document which represents dimension of vectors.
* By using vector calculus, we find the dissimilarity distance between two
* documents.
* First find dot product between two vectors and store in variable Dvec
* (Summation of products of each word frequencies from two documents).
* A high dot product value denotes there are words that are lot of things in
* common.
* Now to find scaling variant/ scaling similarity we have to divide dvec with
* dot product of length(no. of unique words) of the two vectors.
* Then find arc cos of the final result.
* however, that this is not a proper distance metric in a mathematical sense as it
* does not have the triangle inequality property and it violates the coincidence axiom.

**Advantages**

* The vector space model has the following advantages over the Standard Boolean model:
* Simple model based on linear algebra
* Term weights not binary
* Allows computing a continuous degree of similarity between queries and documents
* Allows ranking documents according to their possible relevance
* Allows partial matching

1. **Cosine similarity with tf-idf transformation** :

* tf-idf stands for term frequency and inverse document frequency. A survey conducted in 2015 showed that 83% of text-based recommender systems in digital libraries use tf–idf.
* As typical vanilla cosine document distance model does not rank each words according to its importance in the document, it simply works on all words irrespective of the importance of the words and does not exclude repeated words.
* tf-idf transformation works on bag of words collected from the documents and rank them in respect of their importance played in that document.
* Generally it gives importance to noun words.
* First we have to find word document vector.
* Then for individual word we have to find idf value. To find idf value for each word we have to find log of (no. of documents divided by no. of documents where this word is contained).
* To find tf value for each word we have to divide frequency of the word in a document and total no. of unique words in the document.
* Now multiply tf and idf value of each words to get the transformed matrix of words.
* At last we can use cosine document model on the transformed matrix to find similarity factor.

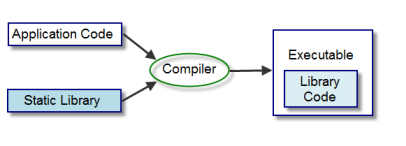
**Advantages**

* Finds words that are relevant in documents.
* Easy to compute
* Finds out importance of keywords or phrase in a document.

**API Design**

It is a computing interface which defines interaction between multiple software intermediaries and it defines kinds of requests that can be made and conventions to follow.

Here in this project I have developed library that will provide a simple and yet powerful API to access this functionality. Libraries are key feature in any programming language to make code portable and this feature has greatly influenced me as well to use in this project.



As speed and size is mainly concerned in this project, hence I chose c++ language out of group of chosen languages because of high efficiency and it supports multiple programming paradigm that is one of the unique flavour found in a language.

There are various ways to design a API but I chose object oriented design because of some of the positive key features such as :

* Encapsulation
* Static and run time polymorphism through Template meta programming
* Greater security
* Robustness
* Abstract data types
* Overloading

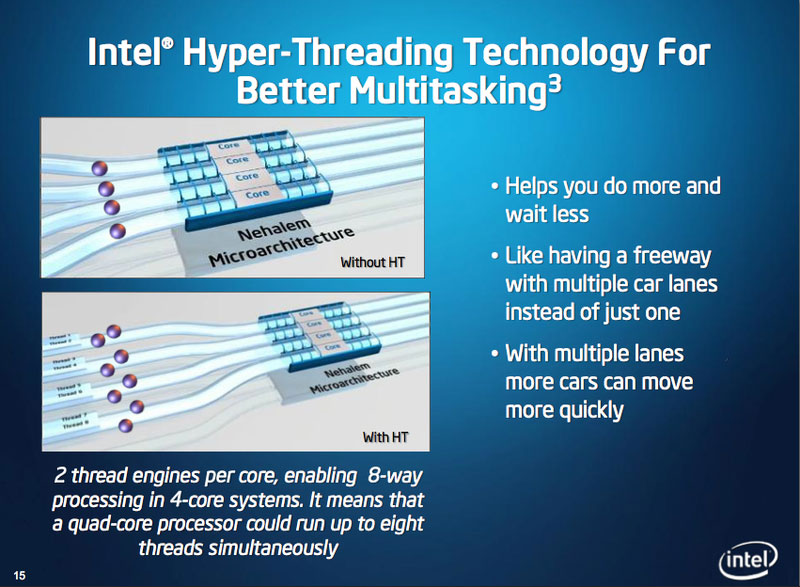
A object oriented approach helped to make this library more dynamic and hence added value to this project success.

Another big reason to choose C++ language over others is because, it gives direct handler to talk with preprocessor and linker.

Design of the library is mentioned in further topics but now other topics are discussed below.

**Multithreading**

When developing this project I have encountered numerous time that even with choosing better algorithm and data structure, the time complexity of WORD function couldn’t be achieved much better. But C++ has excellent multithreading library to handle multithreading efficiently.

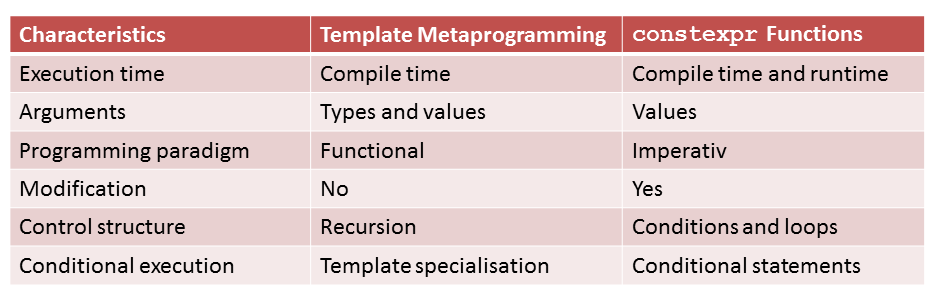


Hence this library will work more efficiently when there are more than one CPU core otherwise the speed can not be improved further. Here POSIX thread has been used to handle repeated function calls thus achieving better time complexity.

**Template Meta Programming**

It is a metaprogramming technique which uses templates and are used by compiler to generate temporary source codes which then merged by the compiler with rest of the source code to generate temporary source code.

I have used this to optimize my codes to generate compile time constants and hence reducing space as much as possible. As it a Turing complete, hence I could possibly use to any computation procedure that is limited by time.



By allowing generating compile time constants and using compiler to do some of the task before will greatly enhance the library time complexity.

Other topics related to this project will be further discussed in other contents briefly.

**PROBLEM DEFINATION AND FEASIBILITY ANALYSIS**

Here problem is divided into two parts :

* IR models
* Library design

*IR models*

Selection of model was a greater challenge for me as time was limited and I had to select minimum models that can effectively address different types of problems in IR.

I have selected two IR models

* Jaccard similarity
* Cosine similarity

Jaccard similarity works on two documents where we find ratio between intersection over union of their word vector.

Cosine similarity works on two vectors where angle between two word vector are measured.

Also I have selected TF-IDF Transformation model with it.

Each models required thorough understanding of key concepts of each type of models where I have noted down few key things :

* Each model work efficiently on vector of words.
* Each use mathematical functions
* Require huge data storage capability in real time.

The problem can be simplified into smaller ones like :

* How to find word vector efficiently ?
* How to compute on word vector efficiently ?
* How to design API without breaking concepts of OO design ?

*Library design*

This is the most important part of the project as the aim is to provide simple yet powerful library to the user and for doing so we need to design this to help code reuse and give users a simple API.

Here flat C++ API or template based API nor data driven API would not work better than a object oriented API design approach. As mentioned before OO API will give rich interface and hiding

capabilities than any other. C++ language offers to use different API style under one roof thus further development in this library has lot of scope to improve.

Here after simplifying the huge problems into smaller ones, these will solved one by one :

* Choosing which programming language to use.
* Dealing with backward compatibility.
* Reuse of functions and abstract data types.
* Handling exceptions.
* Modify codes in library without breaking user’s code.
* Design simple API to minimise user’s interaction.
* Dealing with code migration.
* Dealing with spaghetti code.

These problems are targeted one by one and has been solved at an optimal way and choosing a language for this job is very much crucial job in library design. That is why C++ offers much is the best choice here to design this library. Also it is my personal favourite thing.

Risks are inevitable in every software projects and our goal is to minimise risk to a substantial limit. If library is not properly designed, this could lead to break of code in ways that never imagined before. A user always make mistake, this can be catastrophe if not handled properly and hence care must be taken in every possible way.

Thanks to C++ standard library which paves road to other software architectures to use it efficiently without a worry for backward compatibility. Basic ostream, istream and STL libraries are used in this project. Hence those version of GCC compiler compatible with this libraries, our library is also compatible with it too.

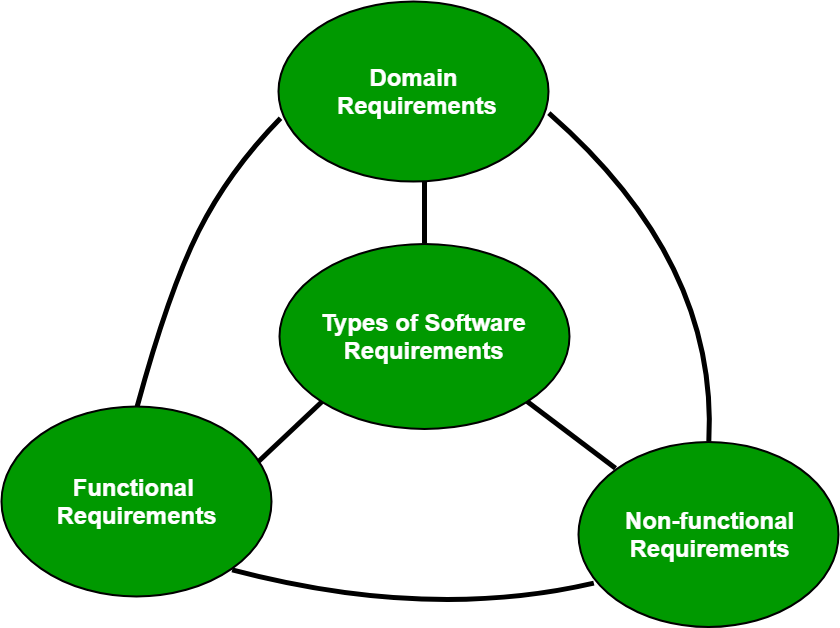
Any further changes to the standard libraries will not break our code and in return will not break users codes who ever using my library to work on their problems.

This library potential are far more greater than other python libraries because :

* C++ is a not a interpreted language, it is precompiled thus faster.
* Famous of system programming hence close to machine level work.
* It supports multi paradigm.
* Allows reusability of codes.
* Game developers love C++ because of raw capability as game require huge computational load.

This library will solve problems in much low powered CPU and thus easy to use in IOT devices too. Thus the project has much scope to develop and it is much independent to work as it requirement are very few.

**SOFTWARE REQUIREMENTS SPECIFICATION**

****

SRS is a document that describes the feature and behaviour of a system. It includes different elements that attempt to define the intended functionality required by the users to satisfy their customers. It generally include technical requirements, constraints, assumptions and acceptance criteria.

**Functional Requirements**

It defines technical details and processing details and other specific functionality that define what the system is supposed to accomplish.

This library is targeted to those users who want fast, reliable C++ library for their IR problems in NLP. This library contains two ADT :

* One ADT to get string input and process it to word vector or bag of words.
* Second ADT is to get this bag of words / word vector and check the contents
* Second ADT must use word vector to use IR models as per their choice of use.

**Hardware**

* Single or multi core processor with hyper threading technology
* Minimum ram to load as per size of document, running processes, compiler.
* Minimum HDD free space of 34 KB to store API library

**Non Functional Requirements**

**Performance**

It defines the performance of the library when in use. Performance are measured by its complexity in time and space and hence target is to lower this complexity as much as possible.

Although time complexity of this project is much promising which is

O( n ) , of time complexity O( 1 ) for space complexity

[ where n = size of the documents which is obvious with documents. ]

**Portability**

Portability defines as how easily this library can be ported to different machines and used thus defines its efficacy. This library is highly portable and is available in Windows . lib file, Linux and Mac as .a file. As most of the devices from small devices to large workstations use variants of Linux, hence this library is effective and highly portable.

**Security**

Various security features are used to in this library to ensure security of unintended / malicious data manipulation during working with it. To enforce security object oriented design has been implemented for this library thus encapsulating data efficiently.

Also private scope is used for the data security so that it cannot be accessed without using the member functions.

Although friend functions are used and has a bad reputation to ruin data security, still carefully design has been implemented by making all the function as const.

Thus const will ensure tighter security if any data member is changed, it will provide compile time error and hence security here is tightened to the limit.

**Maintainability**

Code maintainability is the degree of updating and maintaining library codes to future modify and keep correct copy thus ensuring safety for the project. I have used version control system for this project.

This code can be maintained very easily as I have maintained version controlled system in my Github hence anybody wants to contribute can branch and start contributing to it.

**Reusability**

This library is highly reusable because this API is designed using object oriented style and reusability of this code is very much fuss free, easily derived to create other libraries.

C++ language ensures reusability to a higher degree and lot of system libraries and frameworks are based on this language thus it can easily prove for its reusability.

**Domain Requirements**

These requirements are which are characteristic of a particular category or domain of projects, On this requirements basic needs are shown for project to work efficiently.

Some of the domain requirements like

* Libraries like istream
* ostream
* vector
* unordered map
* utility
* string
* cmath
* thread
* GCC compiler 2011 or higher
* Operating system

These libraries help to achieve abstraction of standard C++ keywords to help coding in this language and eases code non redundancy. This built in libraries are very efficient in time and space complexities and deal with error correction and debugging efficiently.

I’m using GCC C++ compiler of version 2011 to support backward compatibility, otherwise the best choice of compiler would be C++17 as it mostly written in C++17 format and uses various new capabilities like lambda function, threading and function pointers.

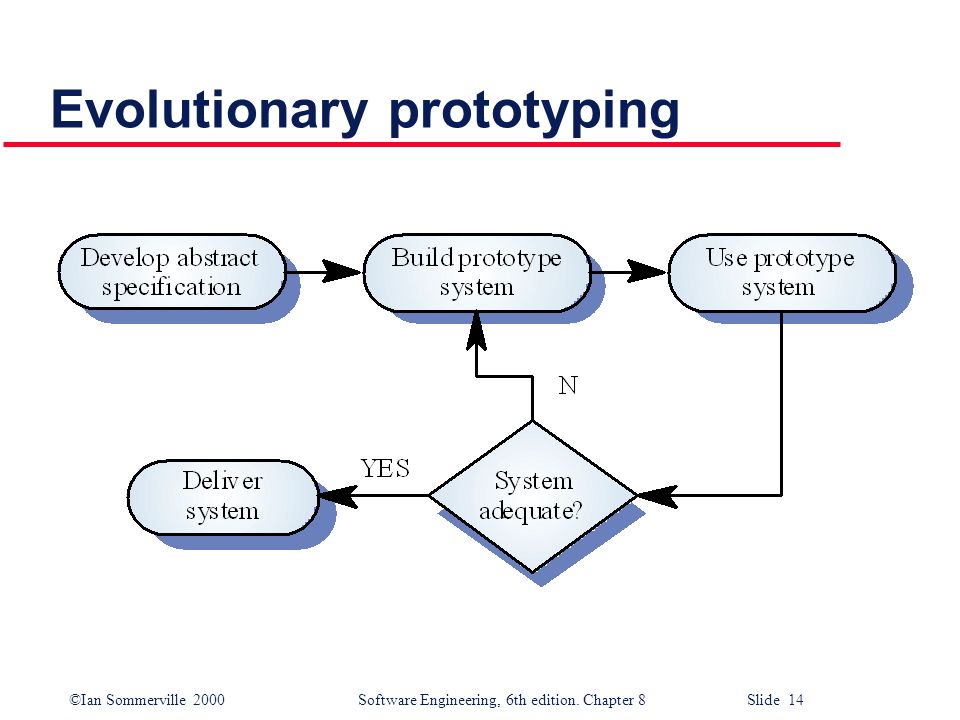
There must be a operating system of your choice like Windows, Mac or Linux.

Also it can run in Tiny Core Linux which is worlds smallest distro with very little core Linux capabilities.

Hence it is capable of running even in small power computers too.

**DEVELOPMENT LIFE CYCLE**

Every project goes through a development cycle where a structured way is used to process every step so that we could design, develop and test high quality software’s. It aims to produce a high quality software that meets expectations, completes within time frame and within estimated costs.

****

Here in this project I have used this model as it was a best fit for this type of project and with my experience in it. Here a prototype developed initially is incrementally refined on the basis of changing / required needs till it finally gets accepted. Here this model offered a better approach which saves time and effort. In this project as it was my first time, I had limited experience in it hence without risking to take big strides, I have developed abstract specification and started building each module one by one and after completion I have thoroughly tested it till requirements are met then again this cycle goes on till the completion of the library. Here this model is famous for building interfaces as it needs constant refinement and initial specification is not concrete.

I have added further steps in it like black box testing and white box testing in this life cycle to develop a custom made life cycle model that suits this project needs perfectly.

This model helped immensely in reducing debugging time and to achieve robust design for the API. There are other steps which revolved around this life cycle model which I have discussed forward.

**SYSTEM ANALYSIS**

In *software engineering*, activities that comprises software engineering as a process in the production of software. The production has four main activities. They are:

* Specification
* Design and implementation
* Validation
* Evolution

Although I am going to specify Specification activity in this topic, other activities like Design will be discussed in next topic, Implementation and Validation ( Testing ) will be covered along with evolution in the consecutive topic .

**Specification**

Software specification is also known as requirement engineering and is defined as the identification of the requirements of the library and the limitations within which the system will operate, develop and can evolve.

**Stages of requirement engineering are :**

**1.Feasibility study**

Here users needs are shortlisted that may help to produce sufficient utility to handle IR problems in NLP. Input that will be provided by the user are sketched and analysed, Based on those inputs our functionality will work and provide solutions to the problem. Based on current available libraries and compilers it was measured that GCC compiler can easily provide all the functionality to develop and address all needs of users. As the requirements of this project is very few because it is almost developed from scratch hence more planning was focussed on designing of API and performance.

**2.Requirements analysis**

Requirements here are analysed and mentioned thoroughly in previous topic.

**3.Requirements specification**

In this stage, documentation of requirements are made and mentioned in previous topic. Later it was thoroughly checked using back tracking method.

**4.Requirements validation**

Here every requirements listed till now are closely validated and checked, though this project uses *Evolutionary Prototyping model* for development. Hence no. of times this stage is revised and the feedback is used to further refine the system.

**SYSTEM DESIGN**

At this stage, the specification starts to become a reality. The library is designed and programming developed. Although the Life cycle model does not guarantee to produce final product in a go, hence system design is revisited numerous time until it became satisfactory quality product.

The stages include the following :

**1.Architectural design**

This includes overall design of the system : main components, sub components and their inter-relationships.

By repeated prototyping, it was concluded that there must be two classes with operations like :

* Word\_to\_vec
* It deals with accepting a string type and storing it. Thus ensuring proper use of the value of this string.
* This type can be changed or modified by user easily with the help of constructor by operator overloading, thus ensuring common C++ operator practices and readability. It also eases a user during development.
* A function that can operate on a string to remove punctuations, stop words, numbers and tokenizing each words into a matrix form.
* Another function to join and produce a single matrix on more than one document together.
* Wordbag
* It accepts a single matrix joined together of more than one document.
* This type can be changed or modified by user easily with the help of constructor by operator overloading, thus ensuring common C++ operator.
* Functions for each implementation of IR model techniques.
* Function for transformation of input matrix.

Each of the classes must uphold strong notion of data security and integrity and for this data members are enclosed in Private scope. Each member functions and friend functions who are dealing with the private data members are enforced using const to read only of private data members.

Blue print of the API will be provided in the header file for easy use of the library and syntactical error handling of the users.

Blue print has numerous pre-processor’s to guide before the compilation of necessary checking’s and ensure non redundancy in codes. Namespace convention is used to help guard the variables use so that they don’t clash with other names in the coding files. It also ensures proper use of dependent standard library files before compilation.

**2.Interface / API design**

It involves the development of interfaces through which components of the system can be used by a user. It is highly appreciated to make the interface as simple as possible and not make the use of the API more complicated.

* Word\_to\_vec
* To declare objects of this class, a users just need to name the class name followed by object name.
* To initialize a class object one can use constructor call during creation of objects or use = operator to copy contents from others. The main motive to not use cin to get input because usually users will use long text essays that they wont type a lengthy passage using cin.



* Word\_vec() function will tokenize and filter words and create a word matrix which is not directly accessible by users. It will be used only by another friend function named generate\_wordbag(). The return type of word\_vec() will be ADT hash matrix of words.



* friend function named generate\_wordbag() will accept two

word\_to\_vec objects and transform it into a word vector. Its return value will be a ADT typically name Hash Matrix of words.

* Wordbag
* To declare objects of this class, a users just need to name the class name followed by object name.
* To initialize a class object one can use constructor call during creation of objects or use = operator to copy contents from others. The main motive to not use cin to get input because users can not produce a complex ADT matrix data for this without hefty coding.
* It accepts matrix data from generate\_wordbag() of word\_to\_vec class.
* It stores matrix data and can be used to generate results on different IR models.



* Jaccard() function will find results on jaccard similarity and return int results.



* cosine() function will find results on cosine similarity and return int results.



* tranform\_tfidf () will transform a copy of private data member and return this transformed matrix.

**3.Component design**

Each component are here designed in object oriented style. So each component is designed a class and within class there are various sub components such as constructors and functions.

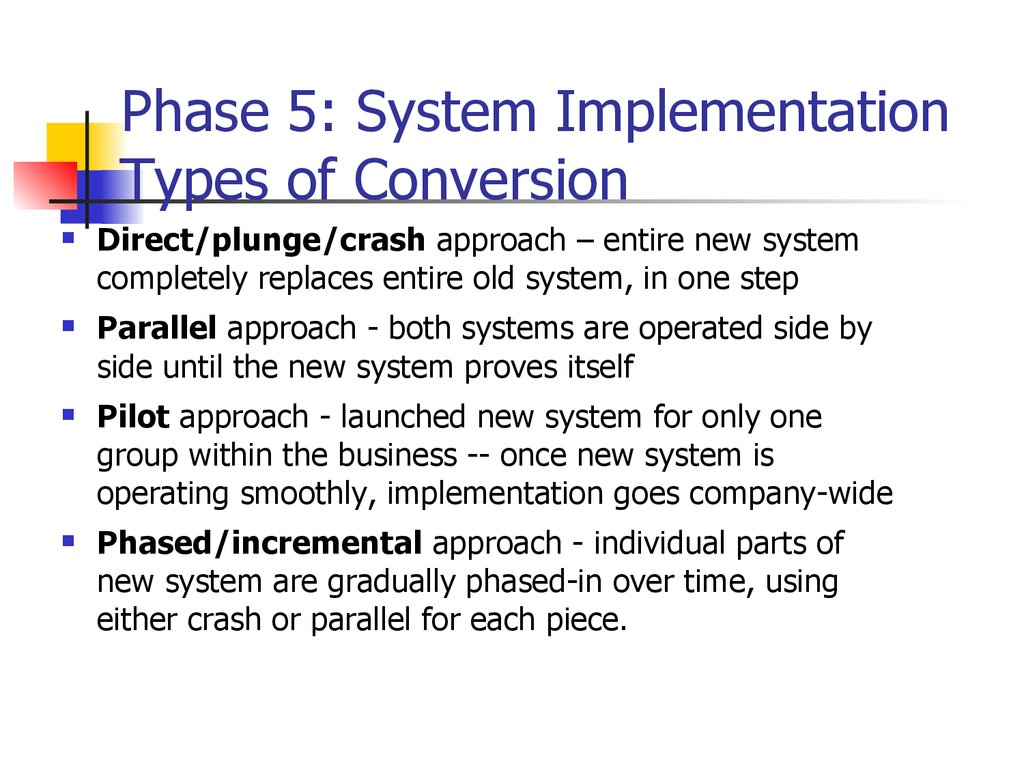
Both the class do not interfere in doing their specific operation. Thus these two classes are independent of each other. However wordbag class can be initialised using word\_to\_vec class component but that is up to the user because they can even use pre ready matrix to initialize wordbag object.

Here *generate\_wordbag()* function has some of the operation where parallel threads operate either fully parallel or virtual parallel ( depends on type of processor ) to speed up operation .

**4.Database design**

Although data driven API’s are designed with storing and processing huge multiverse of data. As no database functionality is required here thus no database design has been implemented.

**IMPLEMENTATION AND TESTING**

****

I have used direct implementation in this project as it was a fresh start of the proposal.

Here Template Metaprogramming is used to derive compile time constants to save memory as much as possible. Although it is a small improvement in memory space and time but it can reduce space complexity of each modules greatly. Here as I have used evolutionary prototype model hence implementing and testing was concurrent and was called again and again till the completion of the project.

**API Testing**

It is the testing of a set of API’s directly and as a part of an integration testing to specify whether they meet requirements for functionality, reliability, performance and security. It needs a driver program or main function to test different aspects of API’s.

* **Unit Testing**

Unit testing is undertaken when a module has been coded and successfully reviewed. This can be done by :

1. **Black Box testing**

Testing, either functional or non functional, without reference to the internal structure of the component modules.

- **Equivalence Class Partitioning :** I have divided input test cases according to different situation and thoroughly checked each results returned by each module, errors produced in handling data and data structure, performance errors.

- **Boundary value analysis :** Some inputs causes severe failure in modules, such inputs are boundary input cases. Hence each boundary input cases has been identified and rectified.

* **White Box Testing**

Testing internal structures of the component modules. There are several white box testing methods as given below:  
  
a)    **Statement coverage**

This strategy aims to design test cases such that every statement in a program is executed at least once.

b)    **Branch coverage**  
Test cases are designed to make each branch condition assume both true and false value, such that each branch is executed once.

c)    **Condition coverage**   
Test cases are designed to make each component of a composite conditional expression assume both true and false values.

d)    **Path coverage**  
Path coverage testing strategy requires us to design such that all linearly independent paths of a program are executed at least once.

* **Integration Testing :**

The objective of integration testing is to check whether the different modules of a program interface with each other properly. Suring this testing, different modules of a system are integrated in a planned manner. After each integration step, the partially integrated system is tested.

In this project the different modules are tested when they are available to realize the full system. Thereby the mixed approach of integration testing is followed.

* **System Testing :**

After integration testing, the system testing is carried out on the fully integrated system. It is done to validate a fully developed system to assure that it meets its requirements as stated in the SRS document. Here alpha testing is carried out on the project. It is a type of system testing which is carried out by the test team within the developing organization. So the project developers carry out the testing themselves to ensure that the requirements have been successfully met.

**CONCLUSION AND FORESEEABLE ENHANCEMENTS**

Although different libraries have their own type of approach to solve this problem, here I tried to focus on two major aspect of library without compromising efficacy and simple API. So we can summarise here that this library works as per expectation and exceeds the expected limit in space complexity. Also this library solves all problems with limited functions currently to deal with most of the problems in IR on NLP.

Here some of the domains in computer science together have been utilised to create a efficient piece of code, backed up with the mathematical computational models. This project will help developers and programmers in C++ who wants to dive in IR in NLP without sacrificing C++ to switch into other language built libraries.

I have taken utmost care to deal with bounded problems but still some of the inbuilt data types like long has limited range and thus I have planned to use boost library to use data types that are dynamically increased on demand to perform functions on ever increasing documents.

This project is yet not complete to fulfil necessary techniques in IR and I hope there are further modification I should do to make it more efficient. Like other IR models must be implemented to be a fully featured NLP Library that can solve every problems efficiently.

Further I am made this project a open source under GNU GPL v3 and wish that other developers to contribute in this and make it a successful library.

**BIBLIOGRAPHY**

#### Martin Reddy. 2011. API Design for C++ (1st. ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

#### Manning, C., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511809071

#### Wikipedia contributors. (2019, December 1). Vector space model. In *Wikipedia, The Free Encyclopedia*. Retrieved 11:55, June 29, 2020, from <https://en.wikipedia.org/w/index.php?title=Vector_space_model&oldid=928749833>

#### Erik Demaine, Ronald Rivest, and Srini Devadas. 6.006 Introduction to Algorithms. Spring 2008. Massachusetts Institute of Technology: MIT OpenCourseWare, [https://ocw.mit.edu](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-spring-2008). License: [Creative Commons BY-NC-SA](https://creativecommons.org/licenses/by-nc-sa/4.0/).

#### David Abrahams and Aleksey Gurtovoy. 2004. C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond (C++ in Depth Series). Addison-Wesley Professional.

#### Thomas Rauber and Gudula Rnger. 2013. Parallel Programming: for Multicore and Cluster Systems (2nd. ed.). Springer Publishing Company, Incorporated

#### Nancy Leveson. 16.355J Software Engineering Concepts. Fall 2005. Massachusetts Institute of Technology: MIT OpenCourseWare, [https://ocw.mit.edu](https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-355j-software-engineering-concepts-fall-2005). License: [Creative Commons BY-NC-SA](https://creativecommons.org/licenses/by-nc-sa/4.0/).

**APPENDIX**

**API Guidelines**

Sourabh Library is a Information Retrieval Library where you can use built in data types and functions that can be used to tokenize strings, find jaccard similarity, cosine similarity and transform using tfidf.

Library is under **namespace aether**

**Example**

#include <sourabh>

int main( )

{

aether :: word\_to\_vec obj;

obj = “Hello world”;

std :: string b = " hello";

aether :: word\_to\_vec cdf = b;

std :: cout << cdf <<’\n’;

aether :: wordbag mat = aether :: generate\_wordbag( obj , cdf );

std :: cout << mat;

std :: cout << “\nJaccard similarity : ” << mat.jaccard();

return 0;

}

**Class Word\_to\_vec** contains

This class accepts a string and uses tokenizing function. It also transform multiple documents to form a bag of word matrix.

**MEMBER VARIABLE**

|  |  |  |
| --- | --- | --- |
| std::string text | Contains built in C++ string type | ( C++11 ) |

**MODIFYING OPERATION**

|  |  |  |
| --- | --- | --- |
| word\_to\_vec () | Class constructor | ( C++11 ) |
| word\_to\_vec ( std::string& ) | Class construction with string reference | ( C++11 ) |
| void operator = ( const std::string& ) | Operator overloading for string assignment | ( C++11 ) |
| ~word\_to\_vec () | Object destructor | ( C++11 ) |

**NON MODIFYING OPERATION**

|  |  |  |
| --- | --- | --- |
| friend std::ostream& operator << ( std::ostream& i, word\_to\_vec& j ) | Outstream operator | ( C++11 ) |
| friend std::unordered\_map < std::string , std::pair < long double, long double > > generate\_wordbag ( word\_to\_vec& , word\_to\_vec& ) | Friend word bag generating operation | ( C++11 ) |
| void word\_vec (std::unordered\_map< std::string , unsigned long >&) const; | String tokenizer and word vector operation | ( C++11 ) |

**Class Wordbag** contains

This class contains information retrieval techniques and transformation. Object of this class can hold a bag of words matrix on which several operation can be used.

**MEMBER VARIABLE**

|  |  |  |
| --- | --- | --- |
| std::unordered\_map < std::string , std::pair < long double , long double > > \_a | Contains hashmap matrix type | ( C++11 ) |

**MODIFYING OPERATION**

|  |  |  |
| --- | --- | --- |
| wordbag () | Class constructor | ( C++11 ) |
| ~wordbag () | Object destructor | ( C++11 ) |
| void operator = ( const std::unordered\_map < std::string , std::pair < long double, long double > >& ) | Operator overloading for hashmap of matrix assignment | ( C++11 ) |

**NON MODIFYING OPERATION**

|  |  |  |
| --- | --- | --- |
| wordbag ( const std::unordered\_map < std::string , std::pair < long double, long double > >& ) const | String tokenizer and word vector operation | ( C++11 ) |
| double cosine () const | Cosine similarity operation | ( C++11 ) |
| double jaccard () const | Jaccard similarity operation | ( C++11 ) |
| std::unordered\_map < std::string , std::pair < long double, long double > > tranform\_tfidf () const | TFIDF transformation operation | ( C++11 ) |
| friend std::ostream& operator << ( std::ostream& , wordbag& ) | Outstream operator | ( C++11 ) |