A MINI PROJECT REPORT

ON

"Vector Space Model Information Retrieval"

Submitted in the partial fulfillment of the requirements for

The degree of

BACHELOR OF ENGINEERING IN COMPUTER ENGINEERING

By

1) NAVNATH AUTI
2) SAHIL NAZARE
3) BINITDEV PANDEY
4) ADNAN SHAIKH

UNDER THE GUIDANCE OF

Prof. Monali Deshmukh

Department of Computer Engineering
Saraswati College of Engineering, Kharghar, Navi-Mumbai
University of Mumbai

2022-23

Saraswati College of Engineering, Kharghar

Vision:

To be universally accepted as autonomous centre of learning in Engineering Education and Research.

Mission:

- To educate students to become responsible and quality technocrats to fulfil society and industry needs.
- To nurture student's creativity and skills for taking up challenges in all facets of life.

Department of Computer Engineering

Vision:

To be among renowned institution in Computer Engineering Education and Research by developing globally competent graduates.

Mission:

- To produce quality Engineering graduates by imparting quality training, hands on experience and value education.
- To pursue research and new technologies in Computer Engineering and across interdisciplinary areas that extends the scope of Computer Engineering and benefit humanity.
- To provide stimulating learning ambience to enhance innovative ideas, problem solving ability, leadership qualities, team-spirit and ethical responsibilities.

DEPARTMENT OF COMPUTER ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVE'S

1. To embed a strong foundation of Computer Engineering fundamentals to identify, solve, analyze and design real time engineering problems as a professional or entrepreneur for the benefit of society.
2. To motivate and prepare students for lifelong learning & research to manifest global competitiveness.
3. To equip students with communication, teamwork and leadership skills to accept challenges in all the facts of life ethically.

SARASWATI Education Society's
SARASWATI College of Engineering
Learn Live Achieve and Contribute
Kharghar, Navi Mumbai - 410 210.

DEPARTMENT OF COMPUTER ENGINEERING

PROGRAM OUTCOMES

1. Apply the knowledge of Mathematics, Science and Engineering Fundamentals to solve complex Computer Engineering Problems.
2. Identify, formulate and analyze Computer Engineering Problems and derive conclusion using First Principle of Mathematics, Engineering Science and Computer Science.
3. Investigate Complex Computer Engineering problems to find appropriate solution leading to valid conclusion.
4. Design a software System, components, Process to meet specified needs with appropriate attention to health and Safety Standards, Environmental and Societal Considerations.
5. Create, select and apply appropriate techniques, resources and advance Engineering software to analyze tools and design for Computer Engineering Problems.
6. Understand the Impact of Computer Engineering solution on society and environment for Sustainable development.
7. Understand Societal, health, Safety, cultural, Legal issues and Responsibilities relevant to Engineering Profession.
8. Apply Professional ethics, accountability and equity in Engineering Profession.
9. Work Effectively as a member and leader in multidisciplinary team for a common goal.
10. Communicate effectively within a Profession and Society at large.
11. Appropriately incorporate principles of Management and Finance in one's own Work.
12. Identify educational needs and engage in lifelong learning in a Changing World of Technology.

DEPARTMENT OF COMPUTER ENGINEERING

PROGRAMME SPECIFIC OUTCOME

1. Formulate and analyze complex engineering problems in computer engineering (Networking/Big data/ Intelligent Systems/Cloud Computing/Real time systems).
2. Plan and develop efficient, reliable, secure and customized application software using cost effective emerging software tools ethically.

SARASWATI Education Society's
SARASWATI College of Engineering
Learn Live Achieve and Contribute
Kharghar, Navi Mumbai - 410 210.

(Approved by AICTE, recg. By Maharashtra Govt. DTE ,Affiliated to Mumbai University)

PLOT NO. 46/46A, SECTOR NO 5, BEHIND MSEB SUBSTATION, KHARGHAR, NAVI MUMBAI-410210

Tel. : 022-27743706 to 11 * Fax : 022-27743712 * Website: www.sce.edu.in

CERTIFICATE

*This is to certify that the requirements for the project report entitled "**Vector Space Model Information Retrieval**" have been successfully completed by the following students:*

Roll numbers	Name
05	Navnath Auti
46	Sahil Nazare
48	Binitdev Pandey
68	Adnan Shaikh

In partial fulfillment of Sem –VII , **Bachelor of Engineering of Mumbai University in Computer Engineering** of Saraswati college of Engineering , Kharghar during the academic year 2022-23.

Internal Guide **External Examiner**

Prof. Monali Deshmukh

Project Co-ordinator **Head of Department**

Dr. Anjali Dadhich Prof. Sujata Bhairnallykar

Principal
Dr. Manjusha Deshmukh

DECLARATION

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included. I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

1. Navnath Auti
2. Sahil Nazare
3. Binitdev Pandey
4. Adnan Shaikh

Date:

ACKNOWLEDGEMENT

After the completion of this work, words are not enough to express feelings about all those who helped us to reach goal.

It's a great pleasure and moment of immense satisfaction for us to express my profound gratitude to **Project Guide**, **Prof. Monali Deshmukh**, whose constant encouragement enabled us to work enthusiastically. His perpetual motivation, patience and excellent expertise in discussion during progress of the project work have benefited us to an extent, which is beyond expression.

We would also like to give our sincere thanks to **Prof. Sujata Bhairnallykar, Head of Department**, and **Prof. Monali Deshmukh** Internal guide from Department of Computer Engineering, Saraswati college of Engineering, Kharghar, Navi Mumbai, for their guidance, encouragement, and support during a project.

I am thankful to **Dr. Manjusha Deshmukh, Principal,**Saraswati College of Engineering, Kharghar, Navi Mumbai for providing an outstanding academic environment, also for providing the adequate facilities.

Last but not the least we would also like to thank all the staffs of Saraswati college of Engineering (Computer Engineering Department) for their valuable guidance with their interest and valuable suggestions brightened us.

1. Navnath Auti
2. Sahil Nazare
3. Binitdev Pandey
4. Adnan Shaikh

ABSTRACT

The Era of digitalization is incomplete without full integration of computer and human world to bridge this gap without Natural language processing is not Possible. Natural language processing (NLP) combines linguistics and artificial intelligence (AI) to enable computers to understand human or natural language input. The business value of NLP is probably obvious. Social data is often information directly created by human input and this data is unstructured in nature, making it nearly impossible to leverage with standard SQL. NLP can make sense of the unstructured data that is produced by social data sources and help to organize it into a more structured model to support SQL-based queries. NLP opens the door for sophisticated analysis of social data and supports text data mining and other sophisticated analytic functions.

The Project is document retrieval system using Vector space model. This use case is widely used in information retrieval systems. Given a set of documents and search term(s)/query we need to retrieve relevant documents that are similar to the search query.

The Vector-Space Model (VSM) for Information Retrieval represents documents and queries as vectors of weights. Each weight is a measure of the importance of an index term in a document or a query, respectively. The index term weights are computed on the basis of the frequency of the index terms in the document, the query or the collection. At retrieval time, the documents are ranked by the cosine of the angle between the document vectors and the query vector. For each document and query, the cosine of the angle is calculated as the ratio between the inner product between the document vector and the query vector, and the product of the norm of the document vector by the norm of the query vector. The documents are then returned by the system by decreasing cosine.

Table of Contents

List of Figures

CHAPTER 1

INTRODUCTON

1.1 GENERAL

The Project is document retrieval system using Vector space model. This use case is widely used in information retrieval systems. Given a set of documents and search term(s)/query we need to retrieve relevant documents that are similar to the search query. Information retrieval (IR), one of the NLP advanced techniques, is defined to be the science of enhancing the effectiveness of term-based document retrieval. It could be also defined as" finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)." Information retrieval systems (IRS) are frequently engineered, optimized and implemented mainly for English language. However, every Language has some special or common features which could be covered by information retrieval techniques with some enhancement. The IR and related methods -will be discussed later-have been used mainly to search for information within documents stored in certain repository of documents (text collection).

The Vector-Space Model (VSM) for Information Retrieval represents documents and queries as vectors of weights. Each weight is a measure of the importance of an index term in a document or a query, respectively. The index term weights are computed on the basis of the frequency of the index terms in the document, the query or the collection. At retrieval time, the documents are ranked by the cosine of the angle between the document vectors and the query vector. For each document and query, the cosine of the angle is calculated as the ratio between the inner product between the document vector and the query vector, and the product of the norm of the document vector by the norm of the query vector. The documents are then returned by the system by decreasing cosine.

The growth of E-libraries with the advancement in computing technology as a result of growth of Internet use has made Electronic data as the major source for the extraction of useful information for the field of Research. It has created a platform for simulation of decision support systems. The Information Retrieval Systems aims at finding right kind of information from documents, unidentified pieces of information from the information base and also tries to search for hidden relationships among the informations retrieved. It has been seen that information can be stored both in structured and unstructured manner. For Example it can be stored in form of Database systems representing

structuring of data whereas storage of HTML and Text files show unstructured behavior. The art and science of searching for information in documents, searching for documents themselves, searching for metadata which describes documents, or searching within databases for text, sound, images or data is Information Retrieval.

A good IR system should be able to perform quickly and successfully the following process:

♣ Accept a user query

♣ Understand from the user query what the user requires

♣ Search a database for relevant documents

♣ Retrieve the documents to the user, and

♣ Rank them in an order based on the relevance of the document to the users initial query

User Task

The user who is looking for some information in a text collection just has to translate his need to some query and pass it to the information retrieval system, this process implies specifying a set of words which convey the semantics of the information need. In addition, the user of the IR system could look for some document or set of documents by using the browsing mechanism that could be offered by the IR systems. A document is described to be relevant if the user makes a decision that the retrieved information gives him an added value with respect to his personal information need.

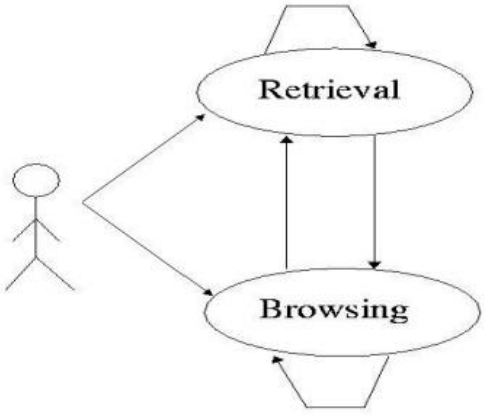

Figure 1.1. User view of IR

1.2 PROBLEM STATEMENT AND OBJECTIVE

1.2.1 PROBLEM STATEMENT

The Project is document retrieval system using Vector space model. This use case is widely used in information retrieval systems. Given a set of documents and search term(s)/query we need to retrieve relevant documents that are similar to the search query.

The Vector-Space Model (VSM) for Information Retrieval represents documents and queries as vectors of weights. Each weight is a measure of the importance of an index term in a document or a query, respectively. The index term weights are computed on the basis of the frequency of the index terms in the document, the query or the collection. At retrieval time, the documents are ranked by the cosine of the angle between the document vectors and the query vector. For each document and query, the cosine of the angle is calculated as the ratio between the inner product between the document vector and the query vector, and the product of the norm of the document vector by the norm of the query vector. The documents are then returned by the system by decreasing cosine.

1.2.2 OBJECTIVE

- Search the required document
- Create a index ranking
- Check the need of information for future
- To give proper and correct information

CHAPTER 2
METHODOLOGY

2.1 ALGORITHMIC DETAILS

Algorithm:

NLTK

NLTK is a toolkit build for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenizing, parse tree visualization, etc. he Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural NLP.

It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets as well as accompanied by a cook book and a book which explains the principles behind the underlying language processing tasks that NLTK supports.

Tokenizing

By **tokenizing**, you can conveniently split up text by word or by sentence. This will allow you to work with smaller pieces of text that are still relatively coherent and meaningful even outside of the context of the rest of the text. It's your first step in turning unstructured data into structured data, which is easier to analyze.

When you're analyzing text, you'll be tokenizing by word and tokenizing by sentence. Here's what both types of tokenization bring to the table:

Tokenizing by word: Words are like the atoms of natural language. They're the smallest unit of meaning that still makes sense on its own. Tokenizing your text by word allows you to identify words that come up particularly often. For example, if you were analyzing a group of job ads, then you might

find that the word “Python” comes up often. That could suggest high demand for Python knowledge, but you’d need to look deeper to know more.

Tokenizing by sentence: When you tokenize by sentence, you can analyze how those words relate to one another and see more context. Are there a lot of negative words around the word “Python” because the hiring manager doesn’t like Python? Are there more terms from the domain of herpetology than the domain of software development, suggesting that you may be dealing with an entirely different kind of python than you were expecting?

Stop Words removal:

When we use the features from a text to model, we will encounter a lot of noise. These are the stop words like the, he, her, etc. Stop words are words that you want to ignore, so you filter them out of your text when you’re processing it. Very common words like 'in', 'is', and 'an' are often used as stop words since they don’t add a lot of meaning to a text in and of themselves. which don’t help us and, just be removed before processing for cleaner processing inside the model. With NLTK we can see all the stop words available in the English language.

from nltk.corpus import stopwords

Stemming:

In our text we may find many words like playing, played, playfully, etc… which have a root word, play all of these convey the same meaning. So we can just extract the root word and remove the rest. Here the root word formed is called ‘stem’ and it is not necessarily that stem needs to exist and have a meaning. Just by committing the suffix and prefix, we generate the stems.

NLTK provides us with PorterStemmer LancasterStemmer and SnowballStemmer packages.

Lemmatization:

We want to extract the base form of the word here. The word extracted here is called Lemma and it is available in the dictionary. We have the WordNet corpus and the lemma generated will be available in this corpus. NLTK provides us with the WordNet Lemmatizer that makes use of the WordNet Database to lookup lemmas of words.

Stemming is much faster than lemmatization as it doesn’t need to lookup in the dictionary and just follows the algorithm to generate the root words.

PANDAS

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users

NUMPY

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software. It contains various features including these important ones:

A powerful N-dimensional array object

Sophisticated (broadcasting) functions

Tools for integrating C/C++ and Fortran code

Useful linear algebra, Fourier transform, and random number capabilities

CORPUS

Each corpus module defines one or more "corpus reader functions", which can be used to read documents from that corpus. These functions take an argument, item, which is used to indicate which document should be read from the corpus:

If item is one of the unique identifiers listed in the corpus module's items variable, then the corresponding document will be loaded from the NLTK corpus package.

If item is a filename, then that file will be read.

Additionally, corpus reader functions can be given lists of item names; in which case, they will return a concatenation of the corresponding documents.

Corpus reader functions are named based on the type of information they return. Some common examples, and their return types, are:

words(): list of str

sents(): list of (list of str)

paras(): list of (list of (list of str))

tagged_words(): list of (str,str) tuple

tagged_sents(): list of (list of (str,str))

tagged_paras(): list of (list of (list of (str,str)))

chunked_sents(): list of (Tree w/ (str,str) leaves)

parsed_sents(): list of (Tree with str leaves)

parsed_paras(): list of (list of (Tree with str leaves))

xml(): A single xml ElementTree

raw(): unprocessed corpus contents

Vector Space Model:

A vector space model is an algebraic model, involving two steps, in first step we represent the text documents into vector of words and in second step we transform to numerical format so that we can apply any text mining techniques such as information retrieval, information extraction,information filtering etc.

Let us understand with an example. consider below statements and a query term. The statements are referred as documents hereafter.

Document 1: Cat runs behind rat
Document 2: Dog runs behind cat
Query: rat

Document vectors representation:

In this step includes breaking each document into words, applying preprocessing steps such as removing stopwords, punctuations, special characters etc. After preprocessing the documents we represent them as vectors of words.
Below is a sample representation of the document vectors.

Document 1: (cat, runs, behind, rat)

Document 2: (Dog, runs, behind, cat)

Query: (rat)

the relevant document to Query = greater of (similarity score between (Document1, Query), similarity score between (Document2, Query)

Next step is to represent the above created vectors of terms to numerical format known as term document matrix.

Term Document Matrix:

A term document matrix is a way of representing documents vectors in a matrix format in which each row represents term vectors across all the documents and columns represent document vectors across all the terms. The cell values frequency counts of each term in corresponding document. If a term is present in a document, then the corresponding cell value contains 1 else if the term is not present in the document, then the cell value contains 0.

After creating the term document matrix, we will calculate term weights for all the terms in the matrix across all the documents. It is also important to calculate the term weightings because we need to find out terms which uniquely define a document.

We should note that a word which occurs in most of the documents might not contribute to represent the document relevance whereas less frequently occurred terms might define document relevance. This can be achieved using a method known as term frequency – inverse document frequency (tf-idf), which gives higher weights to the terms which occurs more in a document but rarely occurs in all other documents, lower weights to the terms which commonly occurs within and across all the documents.

Term weighting

the vector space model (VSM) have to compute some value called ”Term Weight”, this expression is the Crucial part in order to retrieve documents and to rank them based on the relevancy of the document to the used need. There are some main values we have to compute in order to get the term weight.

Term Frequency (TF)

Term Frequency is the standard notion of frequency in natural language processing (NLP); it used to compute the number of times that a term appears in a document. TF expresses the importance of the value in the document and it is widely used since it can be easily calculated inside each document and works well.

Document Frequency (DF)

As the TF represents the importance of the term in the document, the document frequency (df) represents the importance of the term in all documents, it counts the number of documents the term appears in.

Inverse Document Frequency (IDF)

The Inverse Document Frequency which is considered to be a discriminating measure for a term in the text collection. It was proposed in 1972, and has since been widely used. IDF in information retrieval is used to distinguish words that have the same frequency. IDF is defined mathematically as idf = lg10(N/ni),where N = the number of documents in the text collection. ni = is the document frequency, the number of documents that contain term i Tf-idf weighting We now combine the definitions of term frequency (the importance of each index term in the document(tf)) and inverse document frequency(the importance of the index term in the text collection), to produce a composite weight for each term in each document. wij = tfij ∗ idfi = log10(N/ni)

This also called tf-idf scheme, it assigns to word t a weight in document d that is:

• Highest when t occurs many times within a tiny number of documents

• Lower when the term occurs in smaller number of times in a document, or occurs in many documents.

• Lowest when the term exists all documents.

Similarity Measures: cosine similarity

Mathematically, closeness between two vectors is calculated by calculating the cosine angle between two vectors. In similar lines, we can calculate cosine angle between each document vector and the query vector to find its closeness. To find relevant document to the query term , we may calculate the similarity score between each document vector and the query term vector by applying cosine similarity

. Finally, whichever documents having high similarity scores will be considered as relevant documents to the query term.

When we plot the term document matrix, each document vector represents a point in the vector space. In the below example query, Document 1 and Document 2 represent 3 points in the vector space. We can now compare the query with each of the document by calculating the cosine angle between them.

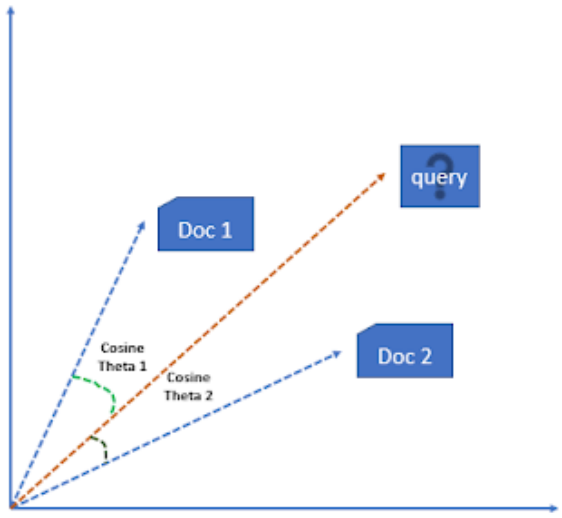

Figure 2.1.1 cosine similarity

Performance Measurement

How could we measure the performance of the information retrieval system? This question lead to create some measurements in order to evaluate the IR system. Precision and Recall are used mainly in order to evaluate NLP systems. In information retrieval system works by running a group of queries against a set of documents and the results retrieved are evaluated using precision, recall and other methods. These evaluation measurements depend on the user decision in order to output there results.

Precision, Recall and others Effectiveness is the word which is highlighted in term of measuring the ability of the system to satisfy the user need. It is useful at this position to introduce a table which illustrate the relation between the relative and non- relatives documents against the retrieved and not retrieved documents.

2.2 HARDWARE AND SOFTWARE REQUIREMENTS

2.2.1 HARDWARE REQUIREMENTS

1. RAM : 4GB RAM
2. Hard Drive : 40 GB Hard Drive
3. Processor : Intel Dual Core Processor

2.2.2 SOFTWARE REQUIREMENTS

1. Anaconda
2. Jupyter Notebook
3. Python
4. Pandas
5. Numpy
6. NLTK

2.3 DESIGN DETAILS:

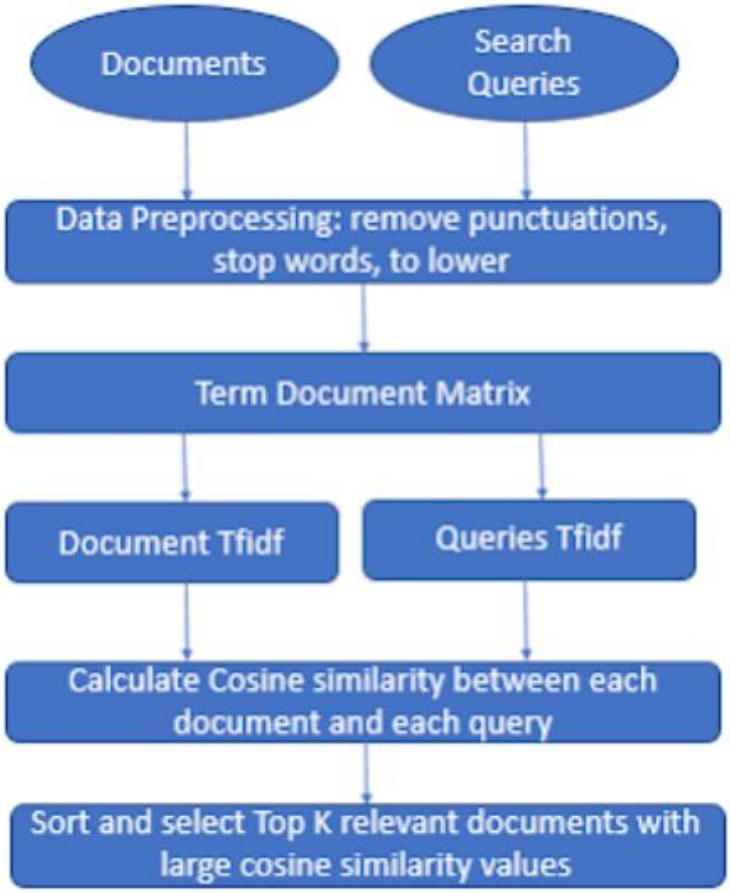

Figure 2.3.1 Retrieval Process

- Load documents and search queries into the environment as list objects.
- Pre-process the data by creating a corpus object with all the documents and query terms, removing stop words, punctuations using tm package.
- Creating a term document matrix with tf-idf weight setting available in TermDocumentMatrix() method.
- Separate the term document matrix into two parts- one containing all the documents with term weights and other containing all the queries with term weights.
- Now calculate cosine similarity between each document and each query.

CHAPTER 3

IMPLEMENTATION AND RESULTS

TF-IDF document retrival with cosine similarity using Vector Space Model

In [1]:
```
import pandas as pd
import numpy as np
import string
import nltk
from nltk.corpus import gutenberg, stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
```

In [2]:
```
gutenberg.fileids()
```

Out[2]:
```
['austen-emma.txt',
 'austen-persuasion.txt',
 'austen-sense.txt',
 'bible-kjv.txt',
 'blake-poems.txt',
 'bryant-stories.txt',
 'burgess-busterbrown.txt',
 'carroll-alice.txt',
 'chesterton-ball.txt',
 'chesterton-brown.txt',
 'chesterton-thursday.txt',
 'edgeworth-parents.txt',
 'melville-moby_dick.txt',
 'milton-paradise.txt',
 'shakespeare-caesar.txt',
 'shakespeare-hamlet.txt',
 'shakespeare-macbeth.txt',
 'whitman-leaves.txt']
```

Statistics of Gutenberg Corpus (Frequency and Expectation)

In [3]:
```
gutenberg_analysis  = [
    [
        len(gutenberg.raw(text)),len(gutenberg.words(text)),
        len(gutenberg.sents(text)),len(set([w.lower() for w in gutenberg.words(text)])),
        text
    ]
    for text in gutenberg.fileids()
]
```

In [4]:
```
for chars,words,sents,vocubs,name in gutenberg_analysis:
    print(f"Text Name: {name} \n\
            Number of characters: {chars}\n\
            Number of words: {words}\n\
            Number of sentences: {sents}\n\
            Number of unique words: {vocubs}\n\
            Average word length: {int(chars/words)}\n\
            Average word in a sentence: {int(words/sents)}\n\
            Average frequency of a word: {int(words/vocubs)}\n\
            ")
```

```
Text Name: austen-emma.txt
            Number of characters: 887071
            Number of words: 192427
            Number of sentences: 7752
            Number of unique words: 7344
            Average word length: 4
            Average word in a sentence: 24
            Average frequency of a word: 26

Text Name: austen-persuasion.txt
            Number of characters: 466292
            Number of words: 98171
            Number of sentences: 3747
            Number of unique words: 5835
            Average word length: 4
            Average word in a sentence: 26
            Average frequency of a word: 16
```

Calculating TF, DF. WF, IDF, WF-IDF Using Vector Space Model

Normalization

TF to WF:

wf = 1 + log(tf) if tf > 0 else 0

DF to IDF:

idf = log(N/df)

WF-IDF

wf-idf = wf x idf

In [5]:
```
stemmer = PorterStemmer()
sw = set(stopwords.words("english") + list(string.punctuation) + list(string.ascii_letters))
freq = {
    "doc_freq": {},
}
for text in gutenberg.fileids():
    doc_name = text.replace("-","_").replace(".txt","")
    freq[doc_name] = {}

    for word in word_tokenize(gutenberg.raw(text)):
        if word in sw:
            continue
        stemmed_word = stemmer.stem(word)
        if stemmed_word in freq[doc_name]:
            freq[doc_name][stemmed_word] += 1
        else:
            freq[doc_name][stemmed_word] = 1
            if stemmed_word in freq["doc_freq"]:
                freq["doc_freq"][stemmed_word] += 1
            else:
```

```
            continue
        stemmed_word = stemmer.stem(word)
        if stemmed_word in freq[doc_name]:
            freq[doc_name][stemmed_word] += 1
        else:
            freq[doc_name][stemmed_word] = 1
            if stemmed_word in freq["doc_freq"]:
                freq["doc_freq"][stemmed_word] += 1
            else:
                freq["doc_freq"][stemmed_word] = 1
```

In [6]:
```
vector_model = pd.DataFrame(freq).fillna(0)
vector_model.head()
```

Out[6]:

	doc_freq	austen_emma	austen_persuasion	austen_sense	bible_kjv	blake_poems	bryant_stories	burgess_busterbrown	carroll_alice	chesterton_ball	chesterton_brown	che
emma	2	855.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
jane	3	301.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
austen	3	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1816	1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
volum	13	3.0	6.0	3.0	2.0	0.0	0.0	0.0	0.0	1.0	2.0	

In [7]:
```
nvm = pd.DataFrame()  #normalized vector space
N = len(vector_model.columns[1:])
nvm["idf"] = np.log10(N/vector_model["doc_freq"])

for column in vector_model.columns[1:]:
    nvm["wf_"+column] = vector_model[column].apply(lambda x: 1+np.log10(x) if x>0 else 0)
    nvm["wf_idf_"+column] = nvm["idf"]*nvm["wf_"+column]
```

Out[8]:

	idf	wf_austen_emma	wf_idf_austen_emma	wf_austen_persuasion	wf_idf_austen_persuasion	wf_austen_sense	wf_idf_austen_sense	wf_bible_kjv	wf_idf_bible_kjv	wf_b
emma	0.954243	3.931966	3.752049	1.000000	0.954243	0.000000	0.000000	0.00000	0.000000	
jane	0.778151	3.478566	2.706851	1.000000	0.778151	1.000000	0.778151	0.00000	0.000000	
austen	0.778151	1.000000	0.778151	1.000000	0.778151	1.000000	0.778151	0.00000	0.000000	
1816	1.255273	1.000000	1.255273	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	
volum	0.141329	1.477121	0.208760	1.778151	0.251305	1.477121	0.208760	1.30103	0.183873	
...	...	...	...	...	...	...	...	...	...	
times'	1.255273	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	
they.	1.255273	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	
germin	1.255273	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	
heart-thud	1.255273	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	
blither	1.255273	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	

36944 rows × 37 columns

In [9]:

```
def cosine_score(q,d):
    L2_q, L2_d = np.linalg.norm(q,ord=2), np.linalg.norm(d,ord=2)
    L2 = np.around(L2_q*L2_d,4)
    dot_product = np.around(np.dot(q,d),4)
    return np.around(dot_product/L2,4) if L2 >0 else dot_product

def query(q):
    qtf = {}
    for word in word_tokenize(q):

        if word in sw:
            continue

        if word in qtf:
            qtf[word] += 1
        else:
            qtf[word] = 1

    qtf = pd.Series(qtf)
    q_idf =  pd.Series({key:(nvm["idf"].loc[key] if key in nvm.index else 0) for key in qtf.index})
    qtf_wf_idf = qtf.apply(lambda x: 1+np.log10(x)) * q_idf
    result = {}

    for doc in vector_model.columns[1:]:
        doc_wf_idf = pd.Series({key:(nvm["wf_idf_"+doc].loc[key] if key in nvm.index else 0) for key in qtf.index})
        result[doc] = cosine_score(qtf_wf_idf,doc_wf_idf)

    return sorted(result.items(),key= lambda x: x[1],reverse=True)
```

In [26]:

```
query("caesar is smart")
```

Out[26]:

```
[('melville_moby_dick', 1.0),
 ('chesterton_thursday', 0.999),
 ('chesterton_brown', 0.9984),
 ('shakespeare_hamlet', 0.9962),
 ('bible_kjv', 0.9762),
 ('edgeworth_parents', 0.9631),
 ('shakespeare_macbeth', 0.9281),
 ('shakespeare_caesar', 0.928),
 ('milton_paradise', 0.3731),
 ('whitman_leaves', 0.3731),
 ('austen_persuasion', 0.3725),
 ('bryant_stories', 0.3725),
 ('chesterton_ball', 0.3725),
 ('austen_sense', 0.3723),
 ('burgess_busterbrown', 0.3721),
 ('austen_emma', 0.0),
 ('blake_poems', 0.0),
 ('carroll_alice', 0.0)]
```

Figure 3.1. Implementation and Result

CHAPTER 4

CONCLUSION AND FUTURE SCOPE

The project is able to retrieve documents based on Vector Space Model Techniques used for efficient retrieval techniques. It is bare fact that each systems has its own strengths and weaknesses what we have sort out in our work for Vector Space Model is the model is Easy to Understand, cheaper to implement considering to the fact that the system should be cost effective i.e. should follow the Space/Time constraint.

We will work in the future to improve the tool in order to enlarge its features, such as covering pdf, html and other file format. Also, give the system the ability to detect the type of device from which the request has been sent in order to handle the request in effective manner and give the user the ability to browse this web tool based on the capabilities of such device. Moreover, we have to make our website secure by limiting access for some features in the website to the anonymous user and allow these features to the granted users only.

CHAPTER 5

REFERENCES

[1] https://alexmoltzau.medium.com/vector-space-for-information-retrieval-d00ad3762210

[2] https://www.sciencedirect.com/topics/computer-science/natural-language-processing

[3] https://pandas.pydata.org/docs/

[4] https://numpy.org/doc/stable/reference/

[5] https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/

[6] https://www.nltk.org/

[7] https://link.springer.com/referenceworkentry/