CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

One of the important NLP applications is Text Summarization, which helps users to manage the vast amount of information available, by condensing documents' content and extracting the most relevant facts or topics included. Summarizing a text means giving a brief statement about the main points of some paragraph or document and dispensing needless details or formalities. Before going deep into the system, we will first describe the need of a tool capable of summarizing large amount of data retaining the main points and concepts of document. In today's Internet age, the information is overwhelmed. The internet stores large amount of data about everything. One may not want to go through from an entire page to get the information of his need. Summarizing a particular document may help a user to get the information of his interest from a huge amount of data.

Summaries can be of forms

- Extractive summaries
- Abstractive summaries

Extractive summaries take important sentences from original document to generate summary. Abstractive summaries do not take sentences from original document rather capture important concepts in the original document and generate new sentences. Abstractive summarization approach is similar to the way that human summarizers follow.

Summarization methods are categorized according to what they generate and how they generate.

- Single document summarization
- Multi-document summarization

A summary can be generated from a single document, known as single document summarization and multiple documents, called multi-document summarization.

Summaries can also be categorized as

- Generic summaries
- Query-Based summaries

Generic summarization systems generate summaries containing main topics of documents. In query-based summarization, the generated summaries contain the sentences that are related to the given queries.

The summarization tools provide the facility to a user to give a document to the tool to generate a summary of a document so that the user may save his time by not reading the entire large document but picking up the concept described in the document through the generated summary.
This proposed project is a summarization tool which is a desktop application that comforts the user eliminating the need of reading large documents. The tool works by taking a single document from user, extracting the important sentences of document and providing a summary to user. In the proposed project, the generated summary is made more readable by replacing the words in the summary with their synonyms. The system replaces those words in the summary with their meanings which are also present in dictionary. Dictionary is maintained by the system.

1.2 OBJECTIVE

- Automatic summarization involves reduces a text file into a passage or paragraph that conveys the main meaning of the text. The searching of important information from a large text file is very difficult job for the users thus to automatic extract the important information or summary of the text file.

- This summary helps the users to reduce time instead of reading the whole text file and it provide quick Information from the large document. In today's world to Extract information from the World Wide Web is very easy. This extracted information is a huge text repository.

- With the rapid growth of the World Wide Web (internet), information overload is becoming a problem for an increasing large number of people. Automatic summarization can be an indispensable solution to reduce the information overload problem on the web.

1.3. LITERATURE REVIEW

Internet has become the biggest source of information all over the world for past few years. Every individual be it a student, business man or any other belonging to any field concern world wide web to get information of any type. This is because the information is just a click away. As in the internet contains information belonging to every field and the data is in such a large amount that many times it is mixed up and becomes very difficult for the reader to pick up the right one. One has to go through the whole document to find out the information needed. There is no such auto summarizer which can generate the summary of digital documents with difficult words replaced with their meanings easy to understand for users' convenience.

1.4 PROBLEM STATEMENT

As the problem of information overload has grown, and as the quantity of data has increased, so has interest in automatic summarization. All the computer users, be it professional or novices user are particularly affected by this predicament. In this present situation, need of auto summarizer is increased.

The problems due to which this tool, Auto-Text Summarizer has been built are described below:

The problem of	No summarizer tool facilitating people having bad understanding of English
Affects	Computer users specifically those using for reading and searching purpose.
the impact of which is	People not willing to use such kind of tools.
a successful solution would be	Provide users with a tool which generate summary in a more interactive way.

Table 1.4.1

1.5 PROPOSED SYSTEM

Auto Text Summarizer will help computer user get rid of this burden of reading each and every word of large documents to find the material of his interest. Our system will provide the user facility to generate the summary of large documents. The tool operates on a single document and provides a summary of the document. The default length of summary is fixed but the user can increase or decrease the number of lines of summary which will be beneficial for him in regard that in how many lines the user can understand the topic and concept of document. Secondly system will be capable of replacing the strong vocabulary words (present in dictionary maintained by system) with their synonyms. In this project the technique will be used for a very limited vocabulary (maximum 50 words) which can be enhanced in the future.

Features

Summarize

- Text in the text area provided.

The system only focuses on the English text in the document.

1.6 SCOPE OF NEW SYSTEM

This project is serviceable for the computer users who are required to go through from a large amount of information frequently. This system will help them to find out the information of interest saving time by eliminating the need of moving through an entire document. This tool also focuses on the people incapable of understanding strong English vocabulary words. The replacement of words (present in dictionary maintained by system) with their synonyms is distinct attribute of this software tool.

This project has a large scope as it has the following features which help in making it easy to use, understand and modify it:

- ➢ Review research literature on text summarization techniques
- ➢ Enumerate existing summarization techniques, algorithms and open-source software solutions
- ➢ Conceive and implement a methodology to score the accuracy/precision of text summarization programs
- ➢ Develop and evaluate the various summarization techniques

Main Points are:-

- Test suite for evaluating text summarization programs

- Technical write-up on the test suite, the various text summarization algorithms and how well they perform
- Lab demo and presentation comparing the various techniques
- Functional code, with comments and documentation

1.7 OUTCOME OF WORK

The outcome of this system generates summary of documents of different types (html document, word document, user specified text, adobe documents after converting it to word document using existing softwares) with the strong vocabulary words (present in dictionary maintained by system) replaced with their synonyms so that extracted summary is relatively easier to understand.

CHAPTER 2

BASICS

In basics we are providing basic knowledge of Information Retrieval , Ranking , Data Base Management System , Knowledge Base System .Components of Information Retrieval will include theoretical and hardware/software components.

2.1 About Information Retrieval

Automated information retrieval (IR) systems were originally developed to help manage the huge scientific literature that has developed since the 1940s. Many university, corporate, and public libraries now use IR systems to provide access to books, journals, and other documents. Commercial IR systems offer databases containing millions of documents in myriad subject areas. Dictionary and encyclopedia databases are now widely available for PCs. IR has been found useful in such disparate areas as office automation and software engineering. Indeed, any discipline that relies on documents to do its work could potentially use and benefit from IR.

This book is about the data structures and algorithms needed to build IR systems. An IR system matches user queries--formal statements of information needs--to documents stored in a database. A document is a data object, usually textual, though it may also contain other types of data such as photographs, graphs, and so on. Often, the documents themselves are not stored directly in the IR system, but are represented in the system by document surrogates. This chapter, for example, is a document and could be stored in its entirety in an IR database. One might instead, however, choose to create a document surrogate for it consisting of the title, author, and abstract. This is typically done for efficiency, that is, to reduce the size of the database and searching time. Document surrogates are also called documents, and in the rest of the book we will use document to denote both documents and document surrogates.

An IR system must support certain basic operations. There must be a way to enter documents into a database, change the documents, and delete them. There must also be some way to search for documents, and present them to a user. As the following chapters illustrate, IR systems vary greatly in the ways they accomplish these tasks.
An information retrieval system is therefore defined here as any device which aids access to documents specified by subject, and the operations associated with it. The documents can be books, journals, reports, atlases, or other records of thought, or any parts of such

records—articles, chapters, sections, tables, diagrams, or even particular words. The retrieval devices can range from a bare list of contents to a large digital computer and its accessories. The operations can range from simple visual scanning to the most detailed programming.

A retrieval system stores units of information. Each unit is linked in the system to specifications of one or more documents or parts of documents—I will call them items. The user specifies particular units of information—specific subjects—and the system is designed to provide him with a knowledge of all relevant items recorded in the system.

A retrieval system can be studied at three levels:

The way in which units of information, and relevant relations between them, are defined in the system. This is the semantic level of subject analysis.

The general structural features of the system considered as a network of units of information linked to each other and to documentary items. This may be called structural analysis.

The physical mechanisms (hardware) in which the structure is embodied.

2.1.1 Types of Information Retrieval

1.Information-Retrieval Systems (IR)

-Search large bodies of information which are not
specifically formatted as formal data bases.
-Web search engine
-Keyword search of a text base
-Typically read-only

2.Database Management Systems (DBMS)

-Relatively small schema
-Large body of homogeneous data
-Minor or no deductive capability

-Extensive formal update capability
-Shared use for both read and write

3.Knowledge-Base Systems (KBS)

-Relatively small body of heterogeneous information
-Significant deductive capability
-Typical use: support of an intelligent application.

2.1.2 Applications of Information Retrieval

1. FAULT PREDICTION: Fault prediction is the process of identifying faulty modulesbefore an actual fault is detected. It permits, for example,engineers to concentrate their testing effort on thefault-containing modules. Such concentration can reducedevelopment costs and increase quality. Predicting fault prone modules has been primarily addressed using softwarequality metrics. For example, the work of Gyimothy,Ferenc, and Siket correlates structural object-orientedmetrics with faults.In contrast, this section considersrecently proposed IR-based solutions to this problem.These include two clever black-box approaches and one more involved technique

2. Handwriting recognition (or HWR) is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-screens and other devices. The image of the written text may be sensed "off line" from a piece of paper by optical scanning (optical character recognition) or intelligent word recognition. Alternatively, the movements of the pen tip may be sensed "on line", for example by a pen-based computer screen surface, a generally easier task as there are more clues available.

Handwriting recognition principally entails optical character recognition. However, a complete handwriting recognition system also handles formatting, performs correct segmentation into characters and finds the most plausible words.

3.Document Indexing: There are several ways to pre-process documents electronically so as to speed up their retrieval. All of these fall under the general term ‘indexing’: an index is a structure that facilitates rapid location of items of interest, an electronic analog of a book’s index. The most widely used technique is word indexing, where the entries (or terms) in the index are individual words in the document (ignoring ‘stop words’—very common and uninteresting words such as ‘the’, ‘an’, ‘of’, etc). Another technique is concept indexing, where one identifies words or phrases and tries to map them to a thesaurus of synonyms as concepts. Therefore, the terms in the index are concept IDs. Several kinds of indexes are created. The global term-frequency index records how many times each distinct term occurs in the entire document collection. The document term-frequency index records how often a particular term occurs in each document. An optional proximity index records the position of individual terms within the document as word, sentence or paragraph offsets. (A proximity index is voluminous, but can allow searches requiring that two or more terms be within the same sentence, or within so many words of each other.) Some issues in indexing are now discussed.

4.Word Indexing: In specific contexts, the list of stop words may need to be expanded. Thus ‘drug’ would be a stop word in a pharmacology-related collection

To reduce the size of the index and to enable broader searches, words may be transformed prior to indexing. One transformation is normalization, which involves lower-case conversion and removal of variations in person or tense. Thus, ‘children’ is normalized to ‘child’, and ‘brought’ to ‘bring’. Another is stemming, which transforms the word by stripping off suffixes or prefixes (using simple pattern-matching rules) to a ‘root’ form, which is not necessarily an actual word in the language. Stemming can be drastic (and its use therefore controversial). Thus, with the widely used Porter stemming algorithm,5 ‘modulation’ and ‘module’ yield the same root, ‘modul’. The two words mean different things (though they were both originally derived from the Latin term ‘modulus’). Conversely, ‘modular’, the adjectival form of ‘module’, stems to itself, rather than to ‘modul’. Several studies of stemming have cast doubt on its utility for English text.

5. Data Mining With IR Technologies: ‘Data mining’ refers to automated discovery of information by electronic processing of data (typically, large data volumes) that may not have been expressly gathered for that purpose. While some authors use the phrase knowledge discovery, ‘knowledge’ is a somewhat more pretentious term than ‘information’. Data mining programs only detect patterns that might possibly be useful in

confirming hypotheses or generating new, possibly interesting hypotheses. This information can only be elevated to the level of 'knowledge' if and when it proves to be useful. Many detected patterns might in fact be self-obvious to someone who is intimately, or even superficially, familiar with the nature of the data that is being mined. An obvious example is mining of medical data, which discovers the pattern, which is 100% specific, that ovarian cancer occurs only in females.

Data mining uses both classical and modern statistical algorithms. (One of the major vendors in the multi-million-dollar data mining industry is the SAS Institute, vendor of the SAS statistics package.) It also makes use of more flexible, but far more computationally intensive approaches that make fewer assumptions about the nature of the data distribution for individual parameters—such as neural networks. However, for special kinds of data, almost any ad hoc analysis can be performed with specially written programs, as long as the researchers are able to justify their approach.

The phrase 'text mining' originated as a marketing term by IR vendors who saw that, when the data being explored is primarily textual, traditional IR technologies such as automatic clustering could be productively deployed to satisfy information hunger. In many cases, 'text mining software' was little more than repackaging of existing IR software. In the genomics area, however, innovative algorithms that could truly be described as mining of text have produced interesting results. We describe a few of these here: this sample is biased by the reviewer's interest in applications that combine the use of PubMed with components of the UMLS, Genbank and other publicly accessible databases.

2.2 Ranking

In 1957 Luhn published a paper proposing a statistical approach to searching literary information. He suggested that "the more two representations agreed in given elements and their distribution, the higher would be the probability of their representing similar information." Maron and Kuhns (1960) went much further by suggesting how to actually weight terms, including some small-scale experiments in term-weighting. The information retrieval research community has continued to develop many models for the ranking technique over the last 30 years (for an overview, see Belkin and Croft [1987]). Although it is not necessary to understand the theoretical models involved in ranking in detail in order to implement a ranking retrieval system, it is helpful to know about them

as they have provided a framework for the vast majority of retrieval experiments that contributed to the development of the ranking techniques used today.

All the experimental results presented in the models are based on using standard test collections and using standard recall and precision measures for evaluation. Results are presented in a roughly chronological order to provide some sense of the development of knowledge about ranking through these experiments.

Ranking models can be divided into two types: those that rank the query against individual documents and those that rank the query against entire sets of related documents. The first type of ranking model, those that rank individual documents against the query, covers several theoretical models, but the principle ones are the vector space model and the probabilistic model.

Assume that a given textual data set uses i unique terms. A document can then be represented by a vector (t1, t2, t3, . . . , tn), where ti has a value of 1 if term i is present, and 0 if term i is absent in the document. A query can be represented in the same manner. Figure shows this representation for a data set with seven unique terms.

The top section of Figure shows the seven terms in this data set. The second section shows a natural language query and its translation into a conceptual vector, with 1's in vector positions of words included in the query, and 0's to indicate a lack of those words. For example, the first "1" indicates the presence of the word "factor," the second "1" indicates the presence of the word "information," the first "0" indicates the absence of the word "help." The third shows a similar conceptual representation of three documents in this data set. To determine which document best matches the query, a simple dot product of the query vector and each document vector is made (left side of the fourth section) and the results used to rank the documents.

It is possible to perform the same operation using weighted vectors as shown in the right side of the bottom section of Figure. In the example, each term is weighted by the total number of times it appears in the record, for example, "factors" appears twice in document 1, "information" appears three times in document 1, and so on. These term-weights could reflect different measures, such as the scarcity of a term in the data set (i.e., "human" probably occurs less frequently than "systems" in a computer science data set), the frequency of a term in the given document (as shown in the example), or some user-specified term-weight. This term-weighting usually provides substantial improvement in the ranking.

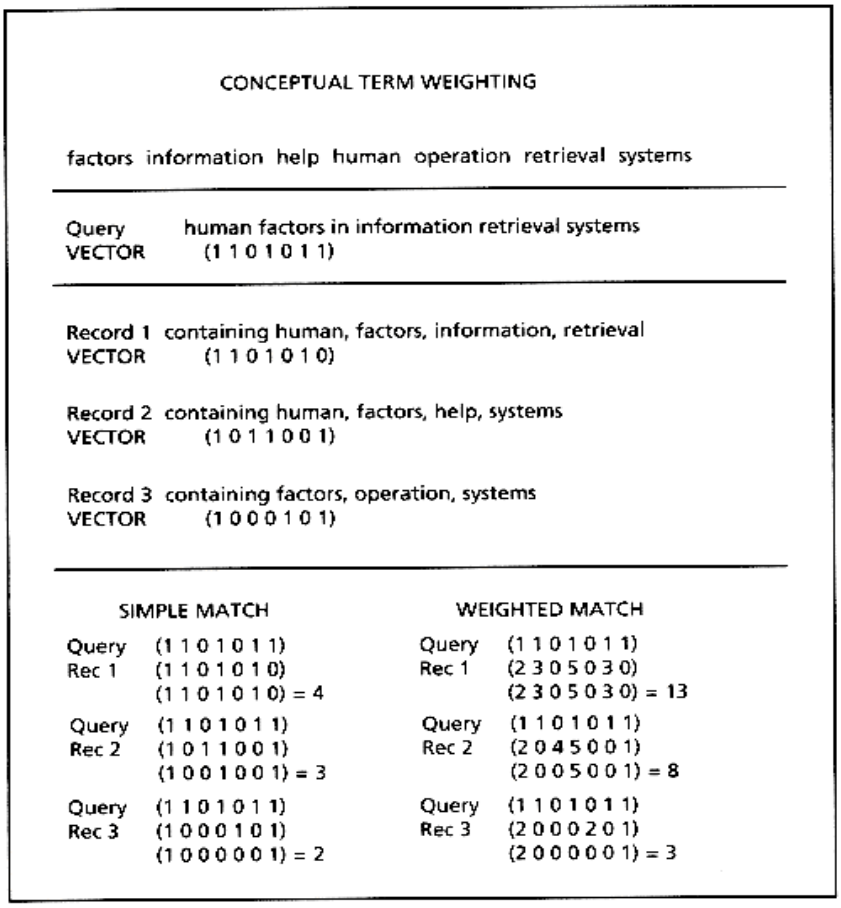


Fig 2.2:Ranking

2.3 Preprocessing:

Remove stop words Stop word referred to counter the obvious fact that many of the words contained in a document do not contribute particularly to the description of the documents content they are less frequent and non-relevant words. For instance, words like “the”, “is” and “and” contribute very little to this description. Therefore it is common to remove these stop words in order to the construction of the document summary, leaving only the content bearing words in the text during processing.

2.4 Stemming :

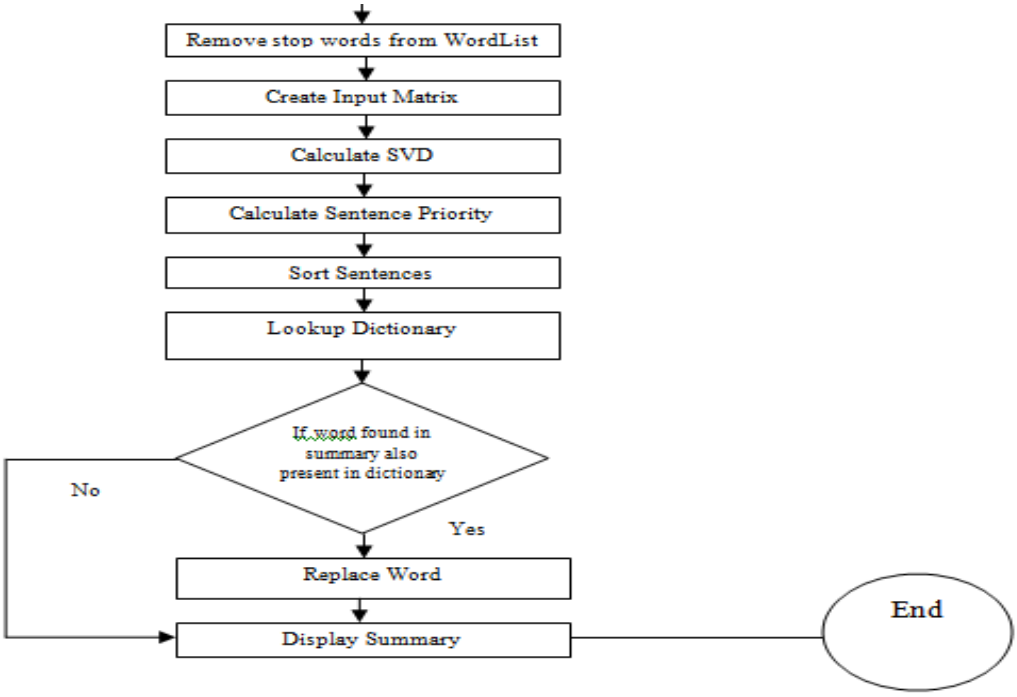
In linguistic morphology and information retrieval, stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. Algorithms for stemming have been studied in computer science since the 1960s. Many search engines treat words with the same stem as synonyms as a kind of query expansion, a process called conflation.Stemming programs are commonly referred to as stemming algorithms or stemmers.

CHAPTER 3

LOGIC

3.1 KNOWLEDGE

1. Flowchart



3.1.1 The Vector Space Model: The sample document and query vectors described can be envisioned as an n-dimensional vector space, where n corresponds to the number of unique terms in the data set. A vector matching operation, based on the cosine correlation used to measure the cosine of the angle between vectors, can then be used to compute the similarity between a document and a query, and documents can be ranked based on that similarity.where tdij = the ith term in the vector for document j , tqik = the ith term in the vector for query k , n = the number of unique terms in the data set. This model has

been used as the basis for many ranking retrieval experiments, in particular the SMART system experiments under Salton and his associates (1968, 1971, 1973, 1981, 1983, 1988). These ranking experiments started in 1964 at Harvard University, moved to Cornell University in 1968, and form a large part of the research in information retrieval. The SMART experiments cover many areas of information retrieval such as relevance feedback, clustering, and experiments with suffixing, synonyms, and phrases. Only those experiments dealing directly with term-weighting and ranking will be discussed here.Early experiments (Salton and Lesk 1968; Salton 1971, p. 143) using the SMART system tested an overlap similarity function against the cosine correlation measure and tried simple term-weighting using the frequency of terms within the documents. These experiments showed that within-document frequency weighting improved performance over no term-weighting (in varying amounts depending on the test collection used). Further, they showed that use of the cosine correlation with frequency term-weighting provided better performance than the overlap similarity because of the automatic inclusion of document length normalization by the cosine similarity function (again, the results varied somewhat depending on the test collection used).A second major set of experiments was done by Salton and Yang (1973) to further develop the term-weighting schemes. Of particular interest in these experiments were the term-weighting schemes relying on term importance within an entire collection rather than only within a given document. Clearly more weight should be given to query terms matching document terms that are rare within a collection. Salton and Yang were able to show significant performance improvement using a term-weighting scheme that combined the within-document frequency weighting with a new term-weighting scheme, the inverted document frequency (IDF; Sparck Jones 1972) that is based on the Zipf distribution of a term within the entire collection (see page 373 for the definition of IDF).A recent paper by Salton and Buckley (1988) summarizes 20 years of SMART experiments in automatic term-weighting by trying 287 distinct combinations of term-weighting assignments, with or without cosine normalization, on six standard collections. Besides confirming that the best document term-weighting is provided by a product of the within-document term frequency and the IDF, normalized by the cosine measure, they show performance improvements using enhanced query term-weighting measures for queries with term frequencies greater than one.

3.1.2 Probabilistic Models: Although a model of probabilistic indexing was proposed and tested by Maron and Kuhns (1960), the major probabilistic model in use today was developed by Robertson and Sparck Jones (1976). This model is based on the premise that terms that appear in previously retrieved relevant documents for a given query should be given a higher weight than if they had not appeared in those relevant

documents. In particular, they presented the following table showing the distribution of term t in relevant and nonrelevant documents for query q.N = the number of documents in the collection.R = the number of relevant documents for query q.n = the number of documents having term t r = the number of relevant documents having term t. They then use this table to derive four formulas that reflect the relative distribution of terms in the relevant and nonrelevant documents, and propose that these formulas be used for term-weighting (the logs are related to actual use of the formulas in term-weighting).Robertson and Sparck Jones also formally derive these formulas, and show that theoretical preference is for F4.Formula F1 had been used by Barkla (1969) for relevance feedback in a SDI service and by Miller (1971) in devising a probabilistic search strategy for Medlars. It was also used by Sparck Jones (1975) in devising optimal performance yardsticks for test collections. Formula F4 (minus the log) is the term precision weighting measure proposed by Yu and Salton (1976).Robertson and Sparck Jones used these four formulas in a series of experiments with the manually indexed Cranfield collection. They did experiments using all the relevance judgments to weight the terms to see what the optimal performance would be, and also used relevance judgments from half the collection to weight the terms for retrieval from the second half of the collection. In both cases, formula F4 was superior (closely followed by F3), with a large drop in performance between the optimal performance and the "predictive" performance, as would be expected. The experimental verification of the theoretical superiority of F4 provided additional weight to the importance of this new model.The use of this theory as a predictive device was further investigated by Sparck Jones (1979a) who used a slightly modified version of F1 and F4 and again got much better results for F4 than for F1, even on a large test collection. Sparck Jones (1979b) tried using this measure (F4 only) in a manner that would mimic a typical on-line session using relevance feedback and found that adding the relevance weighting from only the first couple of relevant documents retrieved by a ranking system still produced performance improvements.Work up to this point using probabilistic indexing required the use of at least a few relevant documents, making this model more closely related to relevance feed-back than to term-weighting schemes of other models. In 1979 Croft and Harper published a paper detailing a series of experiments using probabilistic indexing without any relevance information. Starting with a probabilistic restatement of F4, they assume that all query terms have equal probability of occurring in relevant documents and derive a term-weighting formula that combines a weight based on the number of matching terms and on a term-weighting similar to the IDF measure.where Q = the number of matching terms between document j and query k C = a constant for tuning the similarity functionni = the number of documents having term i in the data set N = the number of documents in the data setExperimental results showed that this term-weighting produced

somewhat better results than the use of the IDF measure alone. Being able to provide different values to C allows this weighting measure to be tailored to various collections. Setting C to 1 ranks the documents by IDF weighting within number of matches, a method that was suitable for the manually indexed Cranfield collection used in this study (because it can be assumed that each matching query term was very significant). C was set much lower in tests with the UKCIS2 collection (Harper 1980) because the terms were assumed to be less accurate, and the documents were very short (consisting of titles only).Croft (1983) expanded his combination weighting scheme to incorporate within-document frequency weights, again using a tuning factor K on these weights to allow tailoring to particular collections. The results show significant improvement over both the IDF weighting alone and the combination weighting, with the scaling factor K playing a large part in tuning the weighting to different collectionswhere Q = the number of matching terms between document j and query kIDFi = the IDF weight for term i in the entire collection (see page 373 for the definition of IDF)where freqij = the frequency of term i in document jK = a constant for adjusting the relative importance of the two weighting schemesmaxfreqj = the maximum frequency of any term in document jThe best value for K proved to be 0.3 for the automatically indexed Cranfield collection, and 0.5 for the NPL collection, confirming that within-document term frequency plays a much smaller role in the NPL collection with its short documents having few repeating terms.

3.1.3 Set-Oriented Ranking Models:

The most well known of the set-oriented models are the clustering models where a query is ranked against a hierarchically grouped set of related documents.

Models based on fuzzy set theory have been proposed (for a summary, see Bookstein [1985]) but have not received enough experimental implementations to be used in practice (except when combined with Boolean queries such as in the P-Norm).

The theory of rough sets has been applied to information retrieval (Srinivasan 1989) but similarly has not been developed far enough to be used in practice.

CHAPTER 4

RANKING ALGORITHM

4.1 Cosine Similarity

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1]. The name derives from the term "direction cosine": in this case, note that unit vectors are maximally "similar" if they're parallel and maximally "dissimilar" if they're orthogonal (perpendicular). This is analogous to the cosine, which is unity (maximum value) when the segments subtend a zero angle and zero (uncorrelated) when the segments are perpendicular.

Note that these bounds apply for any number of dimensions, and cosine similarity is most commonly used in high-dimensional positive spaces. For example, in information retrieval and text mining, each term is notionally assigned a different dimension and a document is characterised by a vector where the value of each dimension corresponds to the number of times that term appears in the document. Cosine similarity then gives a useful measure of how similar two documents are likely to be in terms of their subject matter.

The technique is also used to measure cohesion within clusters in the field of data mining.

One of the reasons for the popularity of cosine similarity is that it is very efficient to evaluate, especially for sparse vectors, as only the non-zero dimensions need to be considered.

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\|_2 \|\mathbf{b}\|_2 \cos\theta$$

Definition

The cosine of two non-zero vectors can be derived by using the Euclidean dot product formula:

Given two vectors of attributes, A and B, the cosine similarity, cos(θ), is represented

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$, where A_i and B_i are components of vector A and B respectively.

using a dot product and magnitude as

The resulting similarity ranges from −1 meaning exactly opposite, to 1 meaning exactly the same, with 0 indicating orthogonality (decorrelation), and in-between values indicating intermediate similarity or dissimilarity.

For text matching, the attribute vectors A and B are usually the term frequency vectors of the documents. The cosine similarity can be seen as a method of normalizing document length during comparison.

In the case of information retrieval, the cosine similarity of two documents will range from 0 to 1, since the term frequencies (tf-idf weights) cannot be negative. The angle between two term frequency vectors cannot be greater than 90°.

If the attribute vectors are normalized by subtracting the vector means (e.g., {\displaystyle A-{\bar {A}}} A-{\bar {A}}), the measure is called centered cosine similarity and is equivalent to the Pearson correlation coefficient.

Angular distance and similarity

The term "cosine similarity" is sometimes used to refer to a different definition of similarity provided below. However the most common use of "cosine similarity" is as defined above and the similarity and distance metrics defined below are referred to as "angular similarity" and "angular distance" respectively. The normalized angle between the vectors is a formal distance metric and can be calculated from the similarity score defined above. This angular distance metric can then be used to compute a similarity function bounded between 0 and 1, inclusive.

$$\text{distance} = \frac{\cos^{-1}(\text{similarity})}{\pi}$$

$$\text{similarity} = 1 - \text{distance}$$

Or, if the vector elements are always positive:

$$\text{distance} = \frac{2 \cdot \cos^{-1}(\text{similarity})}{\pi}$$

$$\text{similarity} = 1 - \text{distance}$$

Although the term "cosine similarity" has been used for this angular distance, the term is used as the cosine of the angle only as a convenient mechanism for calculating the angle itself and is no part of the meaning. The advantage of the angular similarity coefficient is that, when used as a difference coefficient (by subtracting it from 1) the resulting function is a proper distance metric, which is not the case for the first meaning. However, for most uses this is not an important property. For any use where only the relative ordering of similarity or distance within a set of vectors is important, then which function is used is immaterial as the resulting order will be unaffected by the choice.

4.2 Probablistic Model :

The probabilistic retrieval model is based on the Probability Ranking Principle, which states that an information retrieval system is supposed to rank the documents based on their probability of relevance to the query, given all the evidence available [Belkin and Croft 1992]. The principle takes into account that there is uncertainty in the representation of the information need and the documents. There can be a variety of sources of evidence that are used by the probabilistic retrieval methods, and the most common one is the statistical distribution of the terms in both the relevant and non-relevant documents.

We will now describe the state-of-art system developed by Turtle and Croft (1991) that uses Bayesian inference networks to rank documents by using multiple sources of evidence to compute the conditional probability

P(Info need|document) that an information need is satisfied by a given document. An inference network consists of a directed acyclic dependency graph, where edges represent conditional dependency or causal relations between propositions represented by the nodes. The inference network consists of a document network, a concept representation network that represents indexing vocabulary, and a query network representing the information need. The concept representation network is the interface between documents and queries. To compute the rank of a document, the inference network is instantiated and the resulting probabilities are propagated through the network to derive a probability associated with the node representing the information need. These probabilities are used to rank documents.

The statistical approaches have the following strengths:

1) They provide users with a relevance ranking of the retrieved documents. Hence, they enable users to control the output by setting a relevance threshold or by specifying a certain number of documents to display.

2) Queries can be easier to formulate because users do not have to learn a query language and can use natural language.

3) The uncertainty inherent in the choice of query concepts can be represented.

However, the statistical approaches have the following shortcomings:

1) They have a limited expressive power. For example, the NOT operation can not be represented because only positive weights are used. It can be proven that only 2N*N of the 22N possible Boolean queries can be generated by the statistical approaches that use weighted linear sums to rank the documents. This result follows from the analysis of Linear Threshold Networks or Boolean Perceptrons [Anthony and Biggs 1992]. For example, the very common and important Boolean query ((A and B) or (C and D)) can not be represented by a vector space query . Hence, the statistical approaches do not have the expressive power of the Boolean approach.

2) The statistical approach lacks the structure to express important linguistic features such as phrases. Proximity constraints are also difficult to express, a feature that is of great use for experienced searchers.

3) The computation of the relevance scores can be computationally expensive.

4) A ranked linear list provides users with a limited view of the information space and it does not directly suggest how to modify a query if the need arises [Spoerri 1993, Hearst 1994].

5) The queries have to contain a large number of words to improve the retrieval performance. As is the case for the Boolean approach, users are faced with the problem of having to choose the appropriate words that are also used in the relevant documents.

Table below summarizes the advantages and disadvantages that are specific to the vector space and probabilistic model, respectively. This table also shows the formulas that are commonly used to compute the term weights. The two central quantities used are the inverse term frequency in a collection (idf), and the frequencies of a term i in a document j (freq(i,j)). In the probabilistic model, the weight computation also considers how often term appears in the relevant and irrelevant documents, but this presupposes that the

Statistical	**Vector Space**	**Probabilistic**
Motivation	Simplify query formulation Ability to control output	Address uncertainty in query representations
Goal	Rank the output based on Similarity	Probability of Relevance
Methods	Cosine measure	Use of different models
Source	**Query Term Statistics** Vector-Space: • similarity(Q,D) $= \Sigma\ (w_{iq} \times w_{ij})$ / "normalizer" where $w_{iq} = (0.5 + 0.5\ freq_{iq} / maxfreq_q) \times \mathbf{idf}(i)$ $w_{ij} = freq_{ij} \times \mathbf{idf}(i)$ • inverse term freq. in collection $\mathbf{idf}(i) = \log_2 (N - n(i)) / n(i)$. Probabilistic: • term weight $= \log [(r_t / R - r_t) / ((n_t - r_t) / ((N - n_t) - (R - r_t)))]$ ="(hits / misses) / (false alarms/correct misses)" • similarity $_{jk} = \Sigma\ (C + \mathbf{idf}(i)) \times \mathbf{tf}(i,j)$ where $\mathbf{tf}(i,j) = K + (1-K)\ (freq(i,j) / maxfreq(j))$.	
Issues	• How to express NOT ? Proximity searches ? • Limited expressive power • Computationally intensive • Assumes that terms are independent.	• Estimation of needed probabilities • Prior knowledge needed. • Independence assumption • Boolean relations lost. • Which model is best ?
	• Lack of structure to represent important linguistic features • How to better visualize the retrieved set ?	

relevant documents are known or that these frequencies can be reliably estimated

Table 4.2.1 : summarizes the defining characteristics of the statistical retrieval approach, which includes the vector space and the probabilistic model and we list the their key advantages and disadvantages.

If users provide the retrieval system with relevance feedback, then this information is used by the statistical approaches to re compute the weights as follows: the weights of the query terms in the relevant documents are increased, whereas the weights of the query terms that do not appear in the relevant documents are decreased [Salton and Buckley 1990]. There are multiple ways of computing and updating the weights, where each has its advantages and disadvantages. We do not discuss these formulas in more detail, because research on relevance feedback has shown that significant effectiveness improvements can be gained by using quite simple feedback techniques [Salton and Buckley 1990]. Furthermore, what is important to this thesis is that the statistical retrieval approach generates a ranked list, however how this ranking has been computed in detail is immaterial for the purpose of this thesis.

4.3 Fuzzy Set theory Model :

The very basic notion of fuzzy systems is a Fuzzy set . Fuzzy sets are sets whoseelements have degrees of membership. In classical set theory, an element either belongs ordoes not belong to the set. However, fuzzy set theory permits the gradual assessment of themembership of elements in a set; this is described with the help of a membershipfunction valued in the real interval . More precisely, the transition from membership tonon-membership could be gradual rather than abrupt as in the case of Boolean theory .Nevertheless, there has not been a formal basis for how to determine the grade ofmembership. It is a subjective measure that depends on the context, set membership doesnot mean the same thing at the operational level in each and every context. Dubois andPrade have showed in that degree of membership can have one of the followingsemantics: a degree of similarity, a degree of preference, or a degree of uncertainty.

CHAPTER 5

JAVA AND INFORMATION RETRIEVAL

5.1 Java:

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA) meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licences. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Classpath (standard libraries), and IcedTea-Web (browser plugin for applets).

Java platform

Main articles: Java (software platform) and Java virtual machine

One design goal of Java is portability, which means that programs written for the Java platform must run similarly on any combination of hardware and operating system with adequate runtime support. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to architecture-specific machine code. Java bytecode instructions are analogous to machine code, but they are intended to be executed by a virtual machine (VM) written specifically for the host hardware. End users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a web browser for Java applets.

Standard libraries provide a generic way to access host-specific features such as graphics, threading, and networking.

The use of universal bytecode makes porting simple. However, the overhead of interpreting bytecode into machine instructions made interpreted programs almost always run more slowly than native executables. Just-in-time (JIT) compilers that compile bytecodes to machine code during runtime were introduced from an early stage. Java itself is platform-independent, and is adapted to the particular platform it is to run on by a Java virtual machine for it, which translates the Java bytecode into the platform's machine language.

Implementations

Oracle Corporation is the current owner of the official implementation of the Java SE platform, following their acquisition of Sun Microsystems on January 27, 2010. This implementation is based on the original implementation of Java by Sun. The Oracle implementation is available for Microsoft Windows (still works for XP, while only later versions currently officially supported), macOS, Linux and Solaris. Because Java lacks any formal standardization recognized by Ecma International, ISO/IEC, ANSI, or other third-party standards organization, the Oracle implementation is the de facto standard.

The Oracle implementation is packaged into two different distributions: The Java Runtime Environment (JRE) which contains the parts of the Java SE platform required to run Java programs and is intended for end users, and the Java Development Kit (JDK), which is intended for software developers and includes development tools such as the Java compiler, Javadoc, Jar, and a debugger.

OpenJDK is another notable Java SE implementation that is licensed under the GNU GPL. The implementation started when Sun began releasing the Java source code under the GPL. As of Java SE 7, OpenJDK is the official Java reference implementation.

The goal of Java is to make all implementations of Java compatible. Historically, Sun's trademark license for usage of the Java brand insists that all implementations be "compatible". This resulted in a legal dispute with Microsoft after Sun claimed that the Microsoft implementation did not support RMI or JNI and had added platform-specific features of their own. Sun sued in 1997, and, in 2001, won a settlement of US$20 million, as well as a court order enforcing the terms of the license from Sun. As a result, Microsoft no longer ships Java with Windows.

Platform-independent Java is essential to Java EE, and an even more rigorous validation is required to certify an implementation. This environment enables portable server-side applications.

Performance

Main article: Java performance

Programs written in Java have a reputation for being slower and requiring more memory than those written in C++. However, Java programs' execution speed improved significantly with the introduction of just-in-time compilation in 1997/1998 for Java 1.1, the addition of language features supporting better code analysis (such as inner classes, the StringBuilder class, optional assertions, etc.), and optimizations in the Java virtual machine, such as HotSpot becoming the default for Sun's JVM in 2000. With Java 1.5, the performance was improved with the addition of the java.util.concurrent package, including Lock free implementations of the ConcurrentMaps and other multi-core collections, and it was improved further Java 1.6.

Some platforms offer direct hardware support for Java; there are microcontrollers that can run Java in hardware instead of a software Java virtual machine, and some ARM based processors could have hardware support for executing Java bytecode through their Jazelle option, though support has mostly been dropped in current implementations of ARM.

Automatic memory management

Java uses an automatic garbage collector to manage memory in the object lifecycle. The programmer determines when objects are created, and the Java runtime is responsible for recovering the memory once objects are no longer in use. Once no references to an object remain, the unreachable memory becomes eligible to be freed automatically by the garbage collector. Something similar to a memory leak may still occur if a programmer's code holds a reference to an object that is no longer needed, typically when objects that are no longer needed are stored in containers that are still in use. If methods for a nonexistent object are called, a "null pointer exception" is thrown.

One of the ideas behind Java's automatic memory management model is that programmers can be spared the burden of having to perform manual memory management. In some languages, memory for the creation of objects is implicitly allocated on the stack, or explicitly allocated and deallocated from the heap. In the latter case the responsibility of managing memory resides with the programmer. If the program

does not deallocate an object, a memory leak occurs. If the program attempts to access or deallocate memory that has already been deallocated, the result is undefined and difficult to predict, and the program is likely to become unstable and/or crash. This can be partially remedied by the use of smart pointers, but these add overhead and complexity. Note that garbage collection does not prevent "logical" memory leaks, i.e., those where the memory is still referenced but never used.

Garbage collection may happen at any time. Ideally, it will occur when a program is idle. It is guaranteed to be triggered if there is insufficient free memory on the heap to allocate a new object; this can cause a program to stall momentarily. Explicit memory management is not possible in Java.

Java does not support C/C++ style pointer arithmetic, where object addresses and unsigned integers (usually long integers) can be used interchangeably. This allows the garbage collector to relocate referenced objects and ensures type safety and security.

As in C++ and some other object-oriented languages, variables of Java's primitive data types are either stored directly in fields (for objects) or on the stack (for methods) rather than on the heap, as is commonly true for non-primitive data types (but see escape analysis). This was a conscious decision by Java's designers for performance reasons.

Java contains multiple types of garbage collectors. By default,[citation needed] HotSpot uses the parallel scavenge garbage collector. However, there are also several other garbage collectors that can be used to manage the heap. For 90% of applications in Java, the Concurrent Mark-Sweep (CMS) garbage collector is sufficient.[46] Oracle aims to replace CMS with the Garbage-First collector (G1).

What is the Java Programming Environment?

Java is a recently developed, concurrent, class-based, object-oriented programming and runtime environment, consisting of:

A programming language

An API specification

A virtual machine specification

Java has the following characteristics:

Object oriented - Java provides the basic object technology of C++ with some enhancements and some deletions.

Architecture neutral - Java source code is compiled into architecture-independent object code. The object code is interpreted by a Java Virtual Machine (JVM) on the target architecture.

Portable - Java implements additional portability standards. For example, ints are always 32-bit, 2's-complemented integers. User interfaces are built through an abstract window system that is readily implemented in Solaris and other operating environments.

Distributed - Java contains extensive TCP/IP networking facilities. Library routines support protocols such as HyperText Transfer Protocol (HTTP) and file transfer protocol (FTP).

Robust - Both the Java compiler and the Java interpreter provide extensive error checking. Java manages all dynamic memory, checks array bounds, and other exceptions.

Secure - Features of C and C++ that often result in illegal memory accesses are not in the Java language. The interpreter also applies several tests to the compiled code to check for illegal code. After these tests, the compiled code causes no operand stack over- or underflows, performs no illegal data conversions, performs only legal object field accesses, and all opcode parameter types are verified as legal.

High performance - Compilation of programs to an architecture independent machine-like language, results in a small efficient interpreter of Java programs. The Java environment also compiles the Java bytecode into native machine code at runtime.

Multithreaded - Multithreading is built into the Java language. It can improve interactive performance by allowing operations, such as loading an image, to be performed while continuing to process user actions.

Dynamic - Java does not link invoked modules until runtime.

Simple - Java is similar to C++, but with most of the more complex features of C and C++ removed.

Java does not provide:

Programmer-controlled dynamic memory

Pointer arithmetic

struct

typedefs

#define

JRE Components

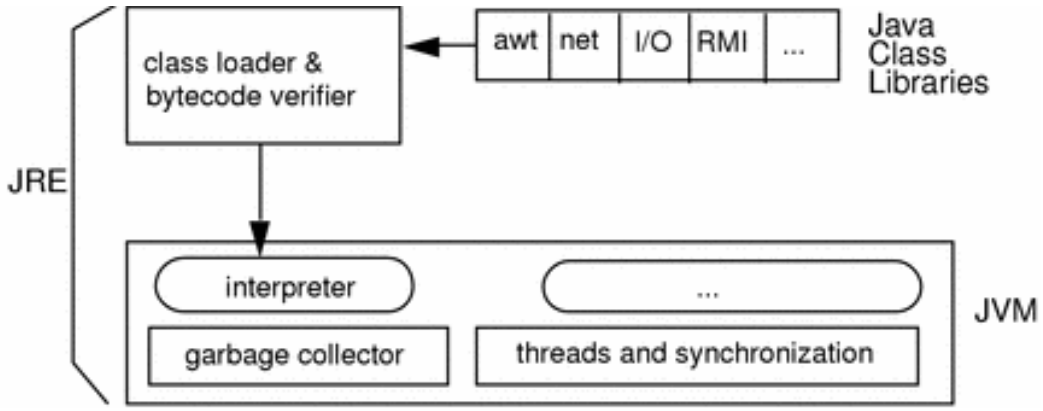
The JRE is the software environment in which programs compiled for a typical JVM implementation can run. The runtime system includes:

Code necessary to run Java programs, dynamically link native methods, manage memory, and handle exceptions

Implementation of the JVM

The following figure shows the JRE and its components, including a typical JVM implementation's various modules and its functional position with respect to the JRE and class libraries.

Figure 5.1 Typical JVM's Implementation: Functional Relationship to JRE and Class



Libraries

JVM

The JVM is an abstract computing machine, having an instruction set that uses memory. Virtual machines are often used to implement a programming language. The JVM is the cornerstone of the Java programming language. It is responsible for Java's cross-platform portability and the small size of its compiled code.

The Solaris JVM is used to execute Java applications. The Java compiler, javac, outputs bytecodes and puts them into a .class file. The JVM then interprets these bytecodes, which can then be executed by any JVM implementation, thus providing Java's cross-platform portability. The next two figures illustrate the traditional compile-time environment and the new portable Java compile-time environment.

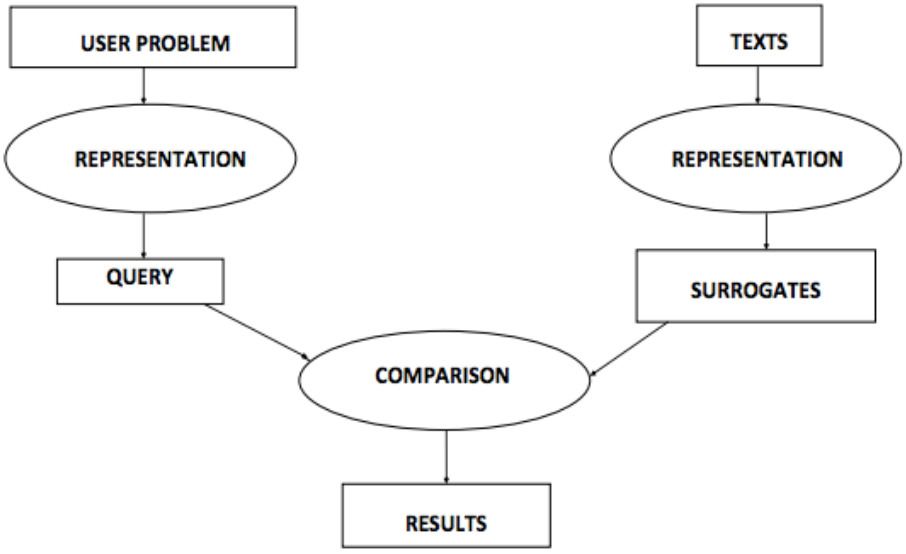
5.2 Information Retrieval

These systems search in the content of a large collection of documents and retrieve the ones that are relevant to a user's demand. The content of a document is generally indexed by terms, and users' queries are also indexed by terms that are used to identify topics of interest. An information retrieval model then assigns a score to each document, that represents its relevance to a query. Terms in documents and queries are usually given specific weights in accordance to their significance. The subject of the internship is to exploit fuzzy logic concepts as well as language modeling approach in order to score and rank documents with respect to a query. Language models define an approach to score documents in IR systems. The interest of using language models in information retrieval comes from the ability to handle structured queries. Fuzzylogic, which is an extension of Boolean logic that takes the values of truth in the real interval [0,1], introduces theoretical foundations to term weighting schemes in information retrieval systems. A theoretical work on fuzzy logic in the database (DB) has been continued for many years by the Pilgrim team (IRISA, Lannion). And it has been recently extended to the field of IR in cooperation with the Texmex team (IRISA, Rennes) with a theoretical and experimental approach. Texmex obtained results showing the validity of the model, and generalizing classical IR vector space model (VSM). The interest of this internship is to explore other tracks to the work done by Texmex, by extending a language model approach using the foundations of fuzzy logic, and consequently to conclude if it corresponds to one of the classical information retrieval systems.This report reviews some of the work done in the scope of information retrieval models that would be interesting to the subject of internship. The report is organized as follows: the

next section, dedicated to information retrieval, consists of three parts; the first part presents IR principals, the second part one of the most popular IR models, namely the vector space model, which explains the concept of weighting terms and scoring documents, the last part talks about language modeling approaches and their application to information retrieval. Section (III), takes a glance at fuzzy logic concepts and possibilistic theory, and presents two approaches to information retrieval related to these

concepts. Some final comments about the interest of applying fuzzy logic within the language model framework are given in the conclusion.An Information Retrieval (IR) System attempts to retrieve from a collection of documents, those relevant documents that correspond to a user's request.

Models of information retrieval systems are characterized by three main components: the representation of documents, the query language, and the matching mechanism. The documents' representation is generated by the indexing process which represents the content of a document as indexed-terms.



Users submit their information needs to the system as queries expressed in the system query language. Then, a matching mechanism evaluates a user's query against the representations of documents and retrieves those documents which are considered to be relevant . Figure represents a traditional view of the information retrieval process. The study of information retrieval models has a long history. And over the decades,

many different types of retrieval models have been proposed and tested. A great diversity of approaches and methodology has been developed. The simplest model as well as the earliest in information retrieval is the Boolean model. In IR Boolean model, users are allowed to specify their information need using a combination of Boolean operations. Therefore, a document would match or not match the query. This means that the notion of document ranking does not exist in a Boolean IR system. Nevertheless, most users expect IR systems to do ranked retrieval. This requires having a mechanism that

determines how relevant a document is for a query .An information retrieval process begins when a user enters a query into the system. Queries are formal statements of information needs, for example search strings in web search engines. In information retrieval a query does not uniquely identify a single object in the collection. Instead, several objects may match the query, perhaps with different degrees of relevancy.

An object is an entity that is represented by information in a content collection or database. User queries are matched against the database information. However, as opposed to classical SQL queries of a database, in information retrieval the results returned may or may not match the query, so results are typically ranked. This ranking of results is a key difference of information retrieval searching compared to database searching.

Depending on the application the data objects may be, for example, text documents, images, audio, mind mapsor videos. Often the documents themselves are not kept or stored directly in the IR system, but are instead represented in the system by document surrogates or metadata.

Most IR systems compute a numeric score on how well each object in the database matches the query, and rank the objects according to this value. The top ranking objects are then shown to the user. The process may then be iterated if the user wishes to refine the query.

CHAPTER 6

AUTOMATIC TEXT SUMMARISATION PROGRAMMING LANGUAGES

6.1 Programming Languages

Introduction to JAVA

Java is a simple and yet powerful object oriented programming language and it is in many respects similar to C++. Java originated at Sun Microsystems, Inc. in 1991. It was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems, Inc. It was developed to provide a platform-independent programming language.Unlike many other programming languages including C and C++ when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by virtual Machine (JVM) on whichever platform it is being run.

Java was designed with a concept of 'write once and run everywhere'. Java Virtual Machine plays the central role in this concept. The JVM is the environment in which Java programs execute. It is a software that is implemented on top of real hardware and operating system. When the source code (.java files) is compiled, it is translated into byte codes and then placed into (.class) files. The JVM executes these bytecodes. So Java byte codes can be thought of as the machine language of the JVM. A JVM can either interpret the bytecode one instruction at a time or the bytecode can be compiled further for the real microprocessor using what is called a just-in-time compiler. The JVM must be implemented on a particular platform before compiled programs can run on that platform.

OBJECT ORIENTED PROGRAMMING

Object Oriented Programming is a method of implementation in which programs are organized as cooperative collection of objects, each of which represents an instance of a class, and whose classes are all members of a hierarchy of classes united via inheritance relationships.

OOP Concepts

Four principles of Object Oriented Programming are

Abstraction
Encapsulation
Inheritance
Polymorphism

Abstraction

Abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer.

Encapsulation

Encapsulation is the process of compartmentalizing the elements of an abstraction that constitute its structure and behavior ; encapsulation serves to separate the contractual interface of an abstraction and its implementation.

* Hides the implementation details of a class.
* Forces the user to use an interface to access data
* Makes the code more maintainable.

Inheritance

Inheritance is the process by which one object acquires the properties of another object.

Polymorphism

Polymorphism is the existence of the classes or methods in different forms or single name denoting different implementations.

Java has powerful features. The following are some of them:-

Since Java is an object oriented programming language it has following features:
Reusability of Code
-Emphasis on data rather than procedure
-Data is hidden and cannot be accessed by external functions
-Objects can communicate with each other through functions
-New data and functions can be easily added
-Simple
-Reusability of Code
-Portable (Platform Independent)
-Distributed
-Robust
-Secure
-High Performance
-Dynamic
-Threaded
-Interpreted

JAVA IS DISTRIBUTED

With extensive set of routines to handle TCP/IP protocols like HTTP and FTP java can open and access the objects across net via URLs.

JAVA IS MULTITHREADED

One of the powerful aspects of the Java language is that it allows multiple threads of execution to run concurrently within the same program A single Java program can have many different threads executing independently and continuously. Multiple Java applets can run on the browser at the same time sharing the CPU time.

JAVA IS SECURE

Java was designed to allow secure execution of code across network. To make Java secure many of the features of C and C++ were eliminated. Java does not use Pointers. Java programs cannot access arbitrary addresses in memory.

GARBAGE COLLECTION

Automatic garbage collection is another great feature of Java with which it prevents inadvertent corruption of memory. Similar to C++, Java has a new operator to allocate memory on the heap for a new object. But it does not use delete operator to free the memory as it is done in C++ to free the memory if the object is no longer needed. It is done automatically with garbage collector.

JAVA APPLICATIONS

Java has evolved from a simple language providing interactive dynamic content for web pages to a predominant enterprise-enabled programming language suitable for developing significant and critical applications. Today, It is used for many types of applications including Web based applications, Financial applications, Gaming applications, embedded systems, Distributed enterprise applications, mobile applications, Image processors, desktop applications and many more. This site outlines the building blocks of java by stating few java examples along with some java tutorials.

CHAPTER 7

AUTOMATIC TEXT SUMMARISATION

Automatic summarization is the process of reducing a text Document with a computer program in order to create a summary that retains the most important points of the original document. As The problem of information overload has grown, and as the quantity of data has increased, so has interest in automatic summarization. It is very difficult for human beings to manually summarize large documents of text. Text Summarization methods can be classified into extractive and abstractive summarization. An extractive summarization method consists of selecting important sentences, paragraphs etc. from the original document and concatenating them into shorter form. The importance of sentences is decided based on statistical and linguistic features of sentences. Extractive methods work by selecting a subset of existing words, phrases, or sentences in the original text to form the summary. The extractive summarization systems are typically based on techniques for sentence extraction and aim to cover the set of sentences that are most important for the overall understanding of a given document.
Automatic summarization is the process of reducing a text document with a computer program in order to create a summary that retains the most important points of the original document. Technologies that can make a coherent summary take into account variables such as length, writing style and syntax. Automatic data summarization is part of machine learning and data mining. The main idea of summarization is to find a representative subset of the data, which contains the information of the entire set. Summarization technologies are used in a large number of sectors in industry today. An example of the use of summarization technology is search engines such as Google. Other examples include document summarization, image collection summarization and video summarization. Document summarization, tries to automatically create a representative summary or abstract of the entire document, by finding the most informative sentences. Similarly, in image summarization the system finds the most representative and important (or salient) images. Similarly, in consumer videos one would want to remove the boring or repetitive scenes, and extract out a much shorter and concise version of the video. This is also important, say for surveillance videos, where one might want to extract only important events in the recorded video, since most part of the video may be uninteresting with nothing going on. As the problem of information overload grows, and as the amount of data increases, the interest in automatic summarization is also increasing.

Generally, there are two approaches to automatic summarization: extraction and abstraction. Extractive methods work by selecting a subset of existing words, phrases, or sentences in the original text to form the summary. In contrast, abstractive methods build

an internal semantic representation and then use natural language generation techniques to create a summary that is closer to what a human might generate. Such a summary might contain words not explicitly present in the original. Research into abstractive methods is an increasingly important and active research area, however due to complexity constraints, research to date has focused primarily on extractive methods. In some application domains, extractive summarization makes more sense. Examples of these include image collection summarization and video summarization.

7.1 Types

Extraction-based summarization

In this summarization task, the automatic system extracts objects from the entire collection, without modifying the objects themselves. Examples of this include keyphrase extraction, where the goal is to select individual words or phrases to "tag" a document, and document summarization, where the goal is to select whole sentences (without modifying them) to create a short paragraph summary. Similarly, in image collection summarization, the system extracts images from the collection without modifying the images themselves.

Abstraction-based summarization

Extraction techniques merely copy the information deemed most important by the system to the summary (for example, key clauses, sentences or paragraphs), while abstraction involves paraphrasing sections of the source document. In general, abstraction can condense a text more strongly than extraction, but the programs that can do this are harder to develop as they require use of natural language generation technology, which itself is a growing field.

While some work has been done in abstractive summarization (creating an abstract synopsis like that of a human), the majority of summarization systems are extractive (selecting a subset of sentences to place in a summary).

Aided summarization

Machine learning techniques from closely related fields such as information retrieval or text mining have been successfully adapted to help automatic summarization.

Apart from Fully Automated Summarizers (FAS), there are systems that aid users with the task of summarization (MAHS = Machine Aided Human Summarization), for example by highlighting candidate passages to be included in the summary, and there are systems that depend on post-processing by a human (HAMS = Human Aided Machine Summarization).

7.2 Applications and systems for summarization

There are broadly two types of extractive summarization tasks depending on what the summarization program focuses on. The first is generic summarization, which focuses on obtaining a generic summary or abstract of the collection (whether documents, or sets of images, or videos, news stories etc.). The second is query relevant summarization, sometimes called query-based summarization, which summarizes objects specific to a query. Summarization systems are able to create both query relevant text summaries and generic machine-generated summaries depending on what the user needs.

An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given document. Sometimes one might be interested in generating a summary from a single source document, while others can use multiple source documents (for example, a cluster of articles on the same topic). This problem is called multi-document summarization. A related application is summarizing news articles. Imagine a system, which automatically pulls together news articles on a given topic (from the web), and concisely represents the latest news as a summary.

Image collection summarization is another application example of automatic summarization. It consists in selecting a representative set of images from a larger set of images. A summary in this context is useful to show the most representative images of results in an image collection exploration system. Video summarization is a related domain, where the system automatically creates a trailer of a long video. This also has applications in consumer or personal videos, where one might want to skip the boring or repetitive actions. Similarly, in surveillance videos, one would want to extract important and suspicious activity, while ignoring all the boring and redundant frames captured.

At a very high level, summarization algorithms try to find subsets of objects (like set of sentences, or a set of images), which cover information of the entire set. This is also called the core-set. These algorithms model notions like diversity, coverage, information and representativeness of the summary. Query based summarization techniques, additionally model for relevance of the summary with the query. Some techniques and algorithms which naturally model summarization problems are TextRank and PageRank, Submodular set function, Determinantal point process, maximal marginal relevance (MMR) etc.

7.2.1 Keyphrase extraction

The task is the following. You are given a piece of text, such as a journal article, and you must produce a list of keywords or key[phrase]s that capture the primary topics discussed in the text. In the case of research articles, many authors provide manually assigned keywords, but most text lacks pre-existing keyphrases. For example, news articles rarely have keyphrases attached, but it would be useful to be able to automatically do so for a number of applications discussed below. Consider the example text from a news article:

"The Army Corps of Engineers, rushing to meet President Bush's promise to protect New Orleans by the start of the 2006 hurricane season, installed defective flood-control pumps last year despite warnings from its own expert that the equipment would fail during a storm, according to documents obtained by The Associated Press".

A keyphrase extractor might select "Army Corps of Engineers", "President Bush", "New Orleans", and "defective flood-control pumps" as keyphrases. These are pulled directly from the text. In contrast, an abstractive keyphrase system would somehow internalize the content and generate keyphrases that do not appear in the text, but more closely resemble what a human might produce, such as "political negligence" or "inadequate protection from floods". Abstraction requires a deep understanding of the text, which makes it difficult for a computer system. Keyphrases have many applications. They can enable document browsing by providing a short summary, improve information retrieval (if documents have keyphrases assigned, a user could search by keyphrase to produce more reliable hits than a full-text search), and be employed in generating index entries for a large text corpus.

Depending on the different literature and the definition of key terms, words or phrases, highly related theme is certainly the Keyword extraction.

Supervised learning approaches

Beginning with the work of Turney,many researchers have approached keyphrase extraction as a supervised machine learning problem. Given a document, we construct an example for each unigram, bigram, and trigram found in the text (though other text units are also possible, as discussed below). We then compute various features describing each example (e.g., does the phrase begin with an upper-case letter?). We assume there are known keyphrases available for a set of training documents. Using the known keyphrases, we can assign positive or negative labels to the examples. Then we learn a classifier that can discriminate between positive and negative examples as a function of the features. Some classifiers make a binary classification for a test example, while others assign a probability of being a keyphrase. For instance, in the above text, we might learn a rule that says phrases with initial capital letters are likely to be keyphrases. After training a learner, we can select keyphrases for test documents in the following manner. We apply the same example-generation strategy to the test documents, then run each example through the learner. We can determine the keyphrases by looking at binary classification decisions or probabilities returned from our learned model. If probabilities are given, a threshold is used to select the keyphrases. Keyphrase extractors are generally evaluated using precision and recall. Precision measures how many of the proposed keyphrases are actually correct. Recall measures how many of the true keyphrases your system proposed. The two measures can be combined in an F-score, which is the harmonic mean of the two ($F = 2PR/(P + R)$). Matches between the proposed keyphrases and the known keyphrases can be checked after stemming or applying some other text normalization.

Designing a supervised keyphrase extraction system involves deciding on several choices (some of these apply to unsupervised, too). The first choice is exactly how to generate examples. Turney and others have used all possible unigrams, bigrams, and trigrams without intervening punctuation and after removing stopwords. Hulth showed that you can get some improvement by selecting examples to be sequences of tokens that match certain patterns of part-of-speech tags. Ideally, the mechanism for generating examples produces all the known labeled keyphrases as candidates, though this is often not the case. For example, if we use only unigrams, bigrams, and trigrams, then we will never be able to extract a known keyphrase containing four words. Thus, recall may suffer. However, generating too many examples can also lead to low precision.

We also need to create features that describe the examples and are informative enough to allow a learning algorithm to discriminate keyphrases from non- keyphrases. Typically features involve various term frequencies (how many times a phrase appears in the current text or in a larger corpus), the length of the example, relative position of the first occurrence, various boolean syntactic features (e.g., contains all caps), etc. The Turney paper used about 12 such features. Hulth uses a reduced set of features, which were found most successful in the KEA (Keyphrase Extraction Algorithm) work derived from Turney's seminal paper.

In the end, the system will need to return a list of keyphrases for a test document, so we need to have a way to limit the number. Ensemble methods (i.e., using votes from several classifiers) have been used to produce numeric scores that can be thresholded to provide a user-provided number of keyphrases. This is the technique used by Turney with C4.5 decision trees. Hulth used a single binary classifier so the learning algorithm implicitly determines the appropriate number.

Once examples and features are created, we need a way to learn to predict keyphrases. Virtually any supervised learning algorithm could be used, such as decision trees, Naive Bayes, and rule induction. In the case of Turney's GenEx algorithm, a genetic algorithm is used to learn parameters for a domain-specific keyphrase extraction algorithm. The extractor follows a series of heuristics to identify keyphrases. The genetic algorithm optimizes parameters for these heuristics with respect to performance on training documents with known key phrases.

Unsupervised approach: TextRank

Another keyphrase extraction algorithm is TextRank. While supervised methods have some nice properties, like being able to produce interpretable rules for what features characterize a keyphrase, they also require a large amount of training data. Many documents with known keyphrases are needed. Furthermore, training on a specific domain tends to customize the extraction process to that domain, so the resulting classifier is not necessarily portable, as some of Turney's results demonstrate. Unsupervised keyphrase extraction removes the need for training data. It approaches the problem from a different angle. Instead of trying to learn explicit features that characterize keyphrases, the TextRank algorithm exploits the structure of the text itself to determine keyphrases that appear "central" to the text in the same way that PageRank selects important Web pages. Recall this is based on the notion of "prestige" or "recommendation" from social networks. In this way, TextRank does not rely on any

previous training data at all, but rather can be run on any arbitrary piece of text, and it can produce output simply based on the text's intrinsic properties. Thus the algorithm is easily portable to new domains and languages.

TextRank is a general purpose graph-based ranking algorithm for NLP. Essentially, it runs PageRank on a graph specially designed for a particular NLP task. For keyphrase extraction, it builds a graph using some set of text units as vertices. Edges are based on some measure of semantic or lexical similarity between the text unit vertices. Unlike PageRank, the edges are typically undirected and can be weighted to reflect a degree of similarity. Once the graph is constructed, it is used to form a stochastic matrix, combined with a damping factor (as in the "random surfer model"), and the ranking over vertices is obtained by finding the eigenvector corresponding to eigenvalue 1 (i.e., the stationary distribution of the random walk on the graph).

The vertices should correspond to what we want to rank. Potentially, we could do something similar to the supervised methods and create a vertex for each unigram, bigram, trigram, etc. However, to keep the graph small, the authors decide to rank individual unigrams in a first step, and then include a second step that merges highly ranked adjacent unigrams to form multi-word phrases. This has a nice side effect of allowing us to produce keyphrases of arbitrary length. For example, if we rank unigrams and find that "advanced", "natural", "language", and "processing" all get high ranks, then we would look at the original text and see that these words appear consecutively and create a final keyphrase using all four together. Note that the unigrams placed in the graph can be filtered by part of speech. The authors found that adjectives and nouns were the best to include. Thus, some linguistic knowledge comes into play in this step.

Edges are created based on word co-occurrence in this application of TextRank. Two vertices are connected by an edge if the unigrams appear within a window of size N in the original text. N is typically around 2–10. Thus, "natural" and "language" might be linked in a text about NLP. "Natural" and "processing" would also be linked because they would both appear in the same string of N words. These edges build on the notion of "text cohesion" and the idea that words that appear near each other are likely related in a meaningful way and "recommend" each other to the reader.

Since this method simply ranks the individual vertices, we need a way to threshold or produce a limited number of keyphrases. The technique chosen is to set a count T to be a user-specified fraction of the total number of vertices in the graph. Then the top T vertices/unigrams are selected based on their stationary probabilities. A post- processing step is then applied to merge adjacent instances of these T unigrams. As a result,

potentially more or less than T final keyphrases will be produced, but the number should be roughly proportional to the length of the original text.

It is not initially clear why applying PageRank to a co-occurrence graph would produce useful keyphrases. One way to think about it is the following. A word that appears multiple times throughout a text may have many different co-occurring neighbors. For example, in a text about machine learning, the unigram "learning" might co-occur with "machine", "supervised", "un-supervised", and "semi-supervised" in four different sentences. Thus, the "learning" vertex would be a central "hub" that connects to these other modifying words. Running PageRank/TextRank on the graph is likely to rank "learning" highly. Similarly, if the text contains the phrase "supervised classification", then there would be an edge between "supervised" and "classification". If "classification" appears several other places and thus has many neighbors, its importance would contribute to the importance of "supervised". If it ends up with a high rank, it will be selected as one of the top T unigrams, along with "learning" and probably "classification". In the final post-processing step, we would then end up with keyphrases "supervised learning" and "supervised classification".

In short, the co-occurrence graph will contain densely connected regions for terms that appear often and in different contexts. A random walk on this graph will have a stationary distribution that assigns large probabilities to the terms in the centers of the clusters. This is similar to densely connected Web pages getting ranked highly by PageRank. This approach has also been used in document summarization, considered below.

7.2.2 Document summarization

Like keyphrase extraction, document summarization aims to identify the essence of a text. The only real difference is that now we are dealing with larger text units—whole sentences instead of words and phrases.

Before getting into the details of some summarization methods, we will mention how summarization systems are typically evaluated. The most common way is using the so-called ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measure. This is a recall-based measure that determines how well a system-generated summary covers the content present in one or more human-generated model summaries known as references. It is recall-based to encourage systems to include all the important topics in the text. Recall can be computed with respect to unigram, bigram, trigram, or 4-gram matching.

For example, ROUGE-1 is computed as division of count of unigrams in reference that appear in system and count of unigrams in reference summary.

If there are multiple references, the ROUGE-1 scores are averaged. Because ROUGE is based only on content overlap, it can determine if the same general concepts are discussed between an automatic summary and a reference summary, but it cannot determine if the result is coherent or the sentences flow together in a sensible manner. High-order n-gram ROUGE measures try to judge fluency to some degree. Note that ROUGE is similar to the BLEU measure for machine translation, but BLEU is precision-based, because translation systems favor accuracy.

A promising line in document summarization is adaptive document/text summarization.The idea of adaptive summarization involves preliminary recognition of document/text genre and subsequent application of summarization algorithms optimized for this genre. First summarizes that perform adaptive summarization have been created.

Supervised learning approaches

Supervised text summarization is very much like supervised keyphrase extraction. Basically, if you have a collection of documents and human-generated summaries for them, you can learn features of sentences that make them good candidates for inclusion in the summary. Features might include the position in the document (i.e., the first few sentences are probably important), the number of words in the sentence, etc. The main difficulty in supervised extractive summarization is that the known summaries must be manually created by extracting sentences so the sentences in an original training document can be labeled as "in summary" or "not in summary". This is not typically how people create summaries, so simply using journal abstracts or existing summaries is usually not sufficient. The sentences in these summaries do not necessarily match up with sentences in the original text, so it would be difficult to assign labels to examples for training. Note, however, that these natural summaries can still be used for evaluation purposes, since ROUGE-1 only cares about unigrams.

Maximum entropy-based summarization

During the DUC 2001 and 2002 evaluation workshops, TNO developed a sentence extraction system for multi-document summarization in the news domain. The system was based on a hybrid system using a naive Bayes classifier and statistical language models for modeling salience. Although the system exhibited good results, the researchers wanted to explore the effectiveness of a maximum entropy (ME) classifier for

the meeting summarization task, as ME is known to be robust against feature dependencies. Maximum entropy has also been applied successfully for summarization in the broadcast news domain.

TextRank and LexRank

The unsupervised approach to summarization is also quite similar in spirit to unsupervised keyphrase extraction and gets around the issue of costly training data. Some unsupervised summarization approaches are based on finding a "centroid" sentence, which is the mean word vector of all the sentences in the document. Then the sentences can be ranked with regard to their similarity to this centroid sentence.

A more principled way to estimate sentence importance is using random walks and eigenvector centrality. LexRank is an algorithm essentially identical to TextRank, and both use this approach for document summarization. The two methods were developed by different groups at the same time, and LexRank simply focused on summarization, but could just as easily be used for keyphrase extraction or any other NLP ranking task.

In both LexRank and TextRank, a graph is constructed by creating a vertex for each sentence in the document.

The edges between sentences are based on some form of semantic similarity or content overlap. While LexRank uses cosine similarity of TF-IDF vectors, TextRank uses a very similar measure based on the number of words two sentences have in common (normalized by the sentences' lengths). The LexRank paper explored using unweighted edges after applying a threshold to the cosine values, but also experimented with using edges with weights equal to the similarity score. TextRank uses continuous similarity scores as weights.

In both algorithms, the sentences are ranked by applying PageRank to the resulting graph. A summary is formed by combining the top ranking sentences, using a threshold or length cutoff to limit the size of the summary.

It is worth noting that TextRank was applied to summarization exactly as described here, while LexRank was used as part of a larger summarization system (MEAD) that combines the LexRank score (stationary probability) with other features like sentence position and length using a linear combination with either user-specified or automatically tuned weights. In this case, some training documents might be needed, though the TextRank results show the additional features are not absolutely necessary.Another

important distinction is that TextRank was used for single document summarization, while LexRank has been applied to multi-document summarization. The task remains the same in both cases—only the number of sentences to choose from has grown. However, when summarizing multiple documents, there is a greater risk of selecting duplicate or highly redundant sentences to place in the same summary. Imagine you have a cluster of news articles on a particular event, and you want to produce one summary. Each article is likely to have many similar sentences, and you would only want to include distinct ideas in the summary. To address this issue, LexRank applies a heuristic post-processing step that builds up a summary by adding sentences in rank order, but discards any sentences that are too similar to ones already placed in the summary. The method used is called Cross-Sentence Information Subsumption (CSIS).

These methods work based on the idea that sentences "recommend" other similar sentences to the reader. Thus, if one sentence is very similar to many others, it will likely be a sentence of great importance. The importance of this sentence also stems from the importance of the sentences "recommending" it. Thus, to get ranked highly and placed in a summary, a sentence must be similar to many sentences that are in turn also similar to many other sentences. This makes intuitive sense and allows the algorithms to be applied to any arbitrary new text. The methods are domain-independent and easily portable. One could imagine the features indicating important sentences in the news domain might vary considerably from the biomedical domain. However, the unsupervised "recommendation"-based approach applies to any domain.

Multi-document summarization

Multi-document summarization is an automatic procedure aimed at extraction of information from multiple texts written about the same topic. Resulting summary report allows individual users, such as professional information consumers, to quickly familiarize themselves with information contained in a large cluster of documents. In such a way, multi-document summarization systems are complementing the news aggregators performing the next step down the road of coping with information overload. Multi-document summarization may also be done in response to a question.

Multi-document summarization creates information reports that are both concise and comprehensive. With different opinions being put together and outlined, every topic is described from multiple perspectives within a single document. While the goal of a brief summary is to simplify information search and cut the time by pointing to the most relevant source documents, comprehensive multi-document summary should itself contain the required information, hence limiting the need for accessing original files to

cases when refinement is required. Automatic summaries present information extracted from multiple sources algorithmically, without any editorial touch or subjective human intervention, thus making it completely unbiased.

7.2.3 Submodular Functions as generic tools for summarization

The idea of a Submodular set function has recently emerged as a powerful modeling tool for various summarization problems. Submodular functions naturally model notions of coverage, information, representation and diversity. Moreover, several important combinatorial optimization problems occur as special instances of submodular optimization. For example, the set cover problem is a special case of submodular optimization, since the set cover function is submodular. The set cover function attempts to find a subset of objects which cover a given set of concepts. For example, in document summarization, one would like the summary to cover all important and relevant concepts in the document. This is an instance of set cover. Similarly, the facility location problem is a special case of submodular functions. The Facility Location function also naturally models coverage and diversity. Another example of a submodular optimization problem is using a Determinantal point process to model diversity. Similarly, the Maximum-Marginal-Relevance procedure can also be seen as an instance of submodular optimization. All these important models encouraging coverage, diversity and information are all submodular. Moreover, submodular functions can be efficiently combined together, and the resulting function is still submodular. Hence, one could combine one submodular function which models diversity, another one which models coverage and use human supervision to learn a right model of a submodular function for the problem.

While submodular functions are fitting problems for summarization, they also admit very efficient algorithms for optimization. For example, a simple greedy algorithm admits a constant factor guarantee.Moreover, the greedy algorithm is extremely simple to implement and can scale to large datasets, which is very important for summarization problems.

Submodular functions have achieved state-of-the-art for almost all summarization problems. For example, work by Lin and Bilmes, 2012 shows that submodular functions achieve the best results to date on DUC-04, DUC-05, DUC-06 and DUC-07 systems for document summarization. Similarly, work by Lin and Bilmes, 2011, shows that many

existing systems for automatic summarization are instances of submodular functions. This was a break through result establishing submodular functions as the right models for summarization problems.

Submodular Functions have also been used for other summarization tasks. Tschiatschek et al., 2014 show that mixtures of submodular functions achieve state-of-the-art results for image collection summarization. Similarly, Bairi et al., 2015 show the utility of submodular functions for summarizing multi-document topic hierarchies. Submodular Functions have also successfully been used for summarizing machine learning datasets.

7.2.4 Evaluation techniques

The most common way to evaluate the informativeness of automatic summaries is to compare them with human-made model summaries.

Evaluation techniques fall into intrinsic and extrinsic, inter-textual and intra-textual.

Intrinsic and extrinsic evaluation

An intrinsic evaluation tests the summarization system in and of itself while an extrinsic evaluation tests the summarization based on how it affects the completion of some other task. Intrinsic evaluations have assessed mainly the coherence and informativeness of summaries. Extrinsic evaluations, on the other hand, have tested the impact of summarization on tasks like relevance assessment, reading comprehension, etc.

Inter-textual and intra-textual

Intra-textual methods assess the output of a specific summarization system, and the inter-textual ones focus on contrastive analysis of outputs of several summarization systems.

Human judgement often has wide variance on what is considered a "good" summary, which means that making the evaluation process automatic is particularly difficult. Manual evaluation can be used, but this is both time and labor-intensive as it requires humans to read not only the summaries but also the source documents. Other issues are those concerning coherence and coverage.

One of the metrics used in NIST's annual Document Understanding Conferences, in which research groups submit their systems for both summarization and translation tasks, is the ROUGE metric (Recall-Oriented Understudy for Gisting Evaluation). It essentially calculates n-gram overlaps between automatically generated summaries and previously-written human summaries. A high level of overlap should indicate a high level of shared concepts between the two summaries. Note that overlap metrics like this are unable to provide any feedback on a summary's coherence. Anaphor resolution remains another problem yet to be fully solved. Similarly, for image summarization, Tschiatschek et al., developed a Visual-ROUGE score which judges the performance of algorithms for image summarization.

Domain specific versus domain independent summarization techniques

Domain independent summarization techniques generally apply sets of general features which can be used to identify information-rich text segments. Recent research focus has drifted to domain-specific summarization techniques that utilize the available knowledge specific to the domain of text. For example, automatic summarization research on medical text generally attempts to utilize the various sources of codified medical knowledge and ontologies.

Evaluating summaries qualitatively

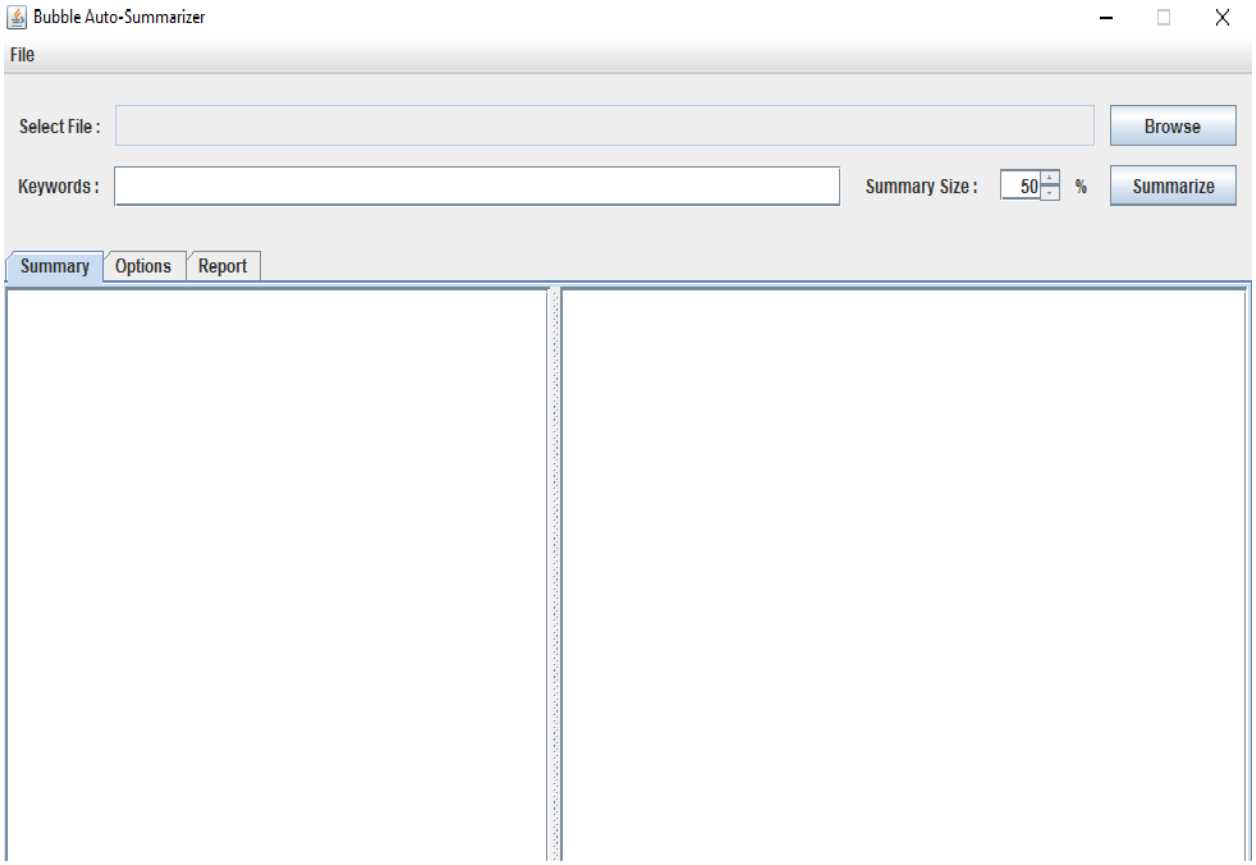
The main drawback of the evaluation systems existing so far is that we need at least one reference summary, and for some methods more than one, to be able to compare automatic summaries with models. This is a hard and expensive task. Much effort has to be done in order to have corpus of texts and their corresponding summaries. Furthermore, for some methods, not only do we need to have human-made summaries available for comparison, but also manual annotation has to be performed in some of them (e.g. SCU in the Pyramid Method). In any case, what the evaluation methods need as an input, is a set of summaries to serve as gold standards and a set of automatic summaries. Moreover, they all perform a quantitative evaluation with regard to different similarity metrics.

`

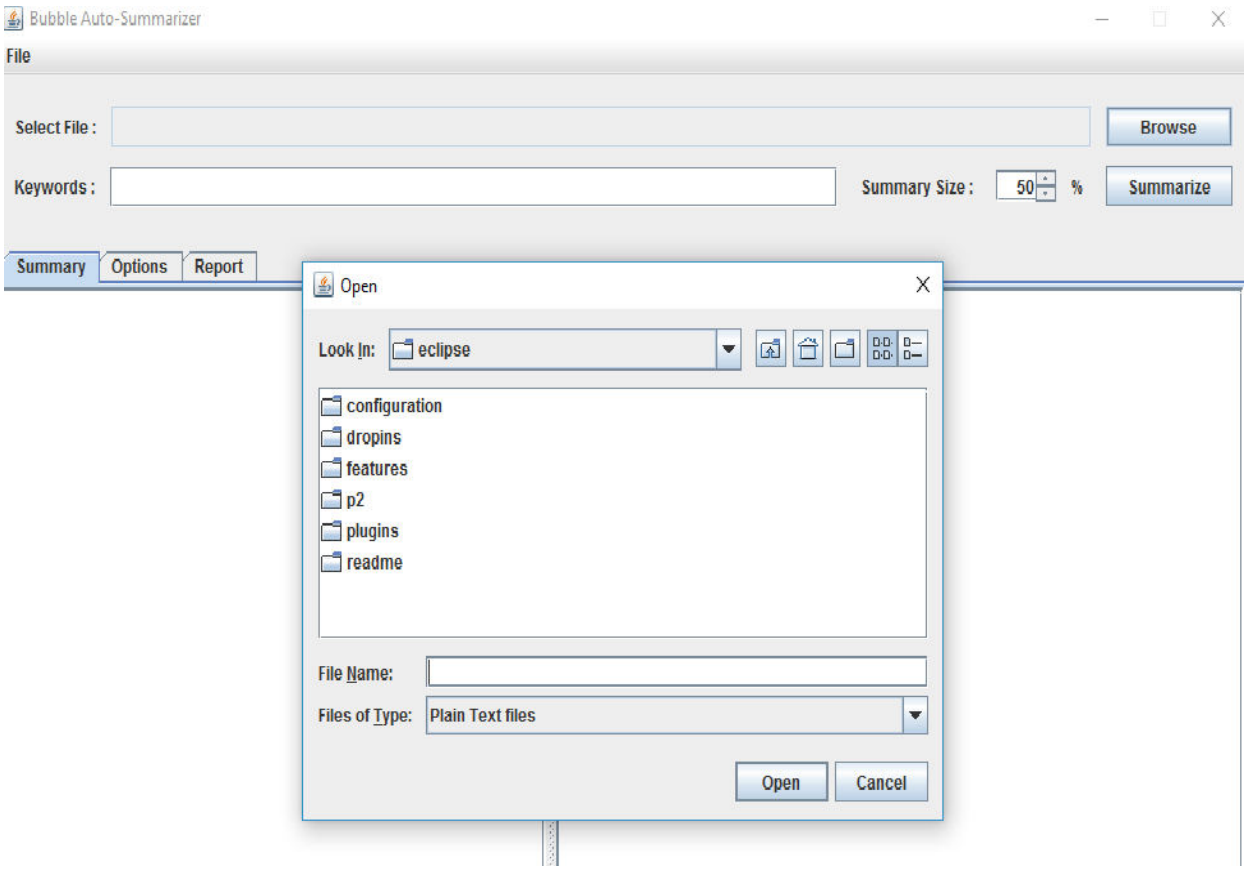
CHAPTER 8

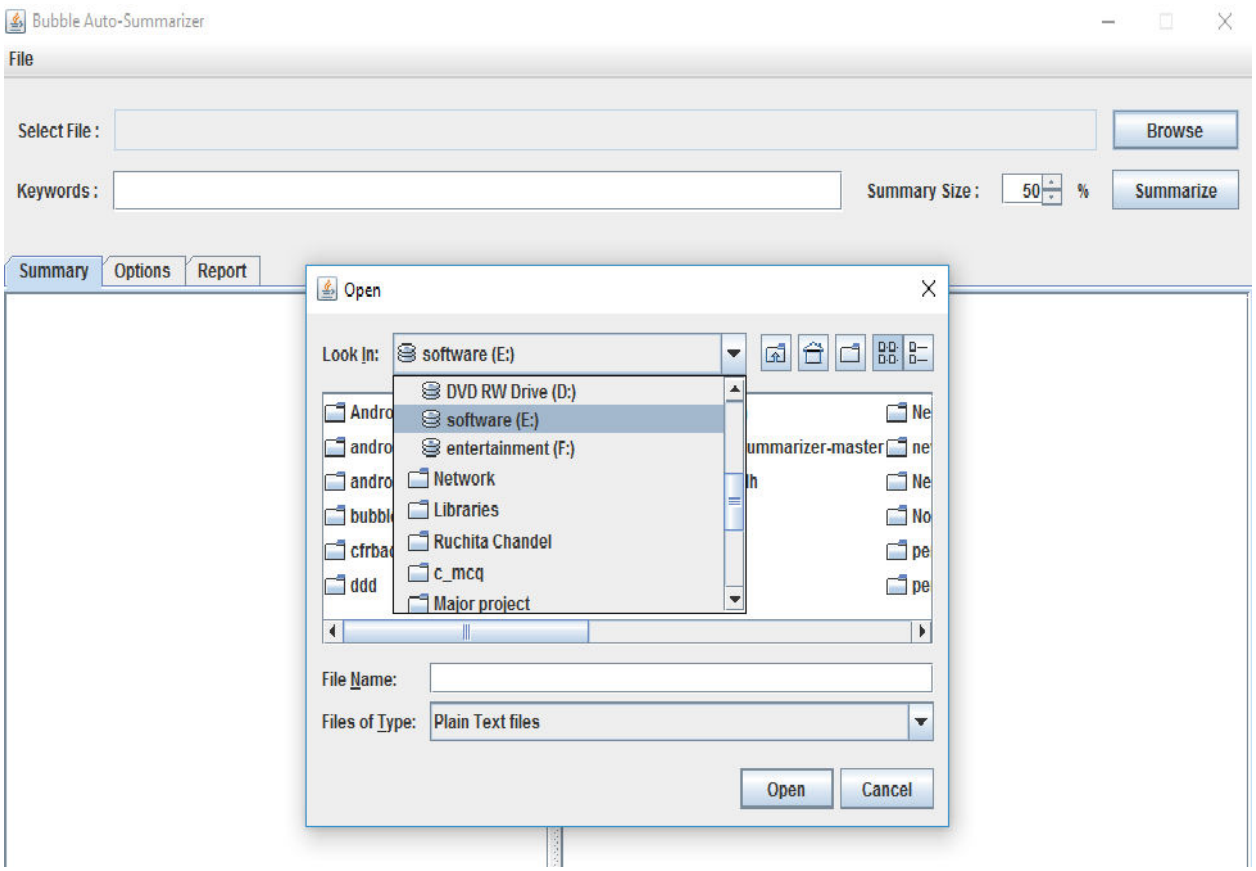
SNAPSHOTS

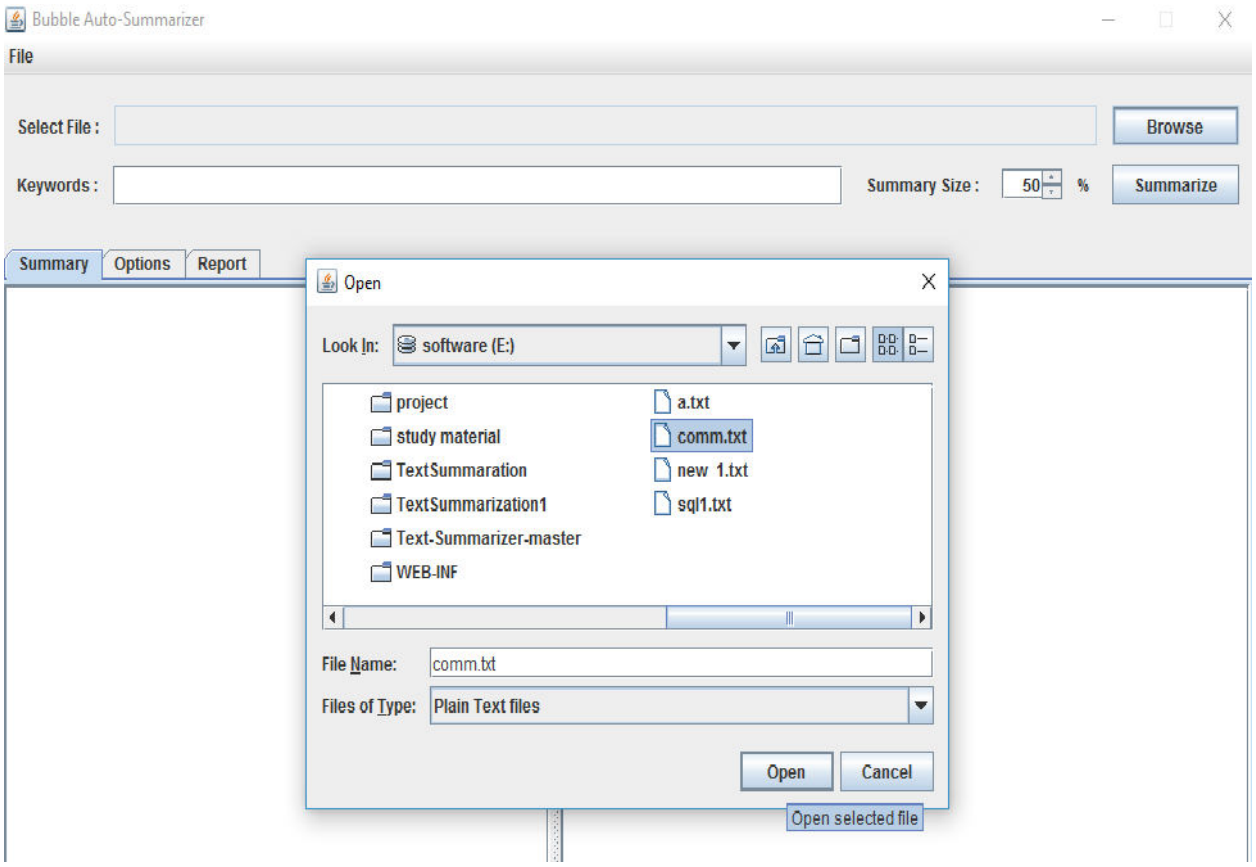
9.1 Snap Shots of Automatic Text Summarization



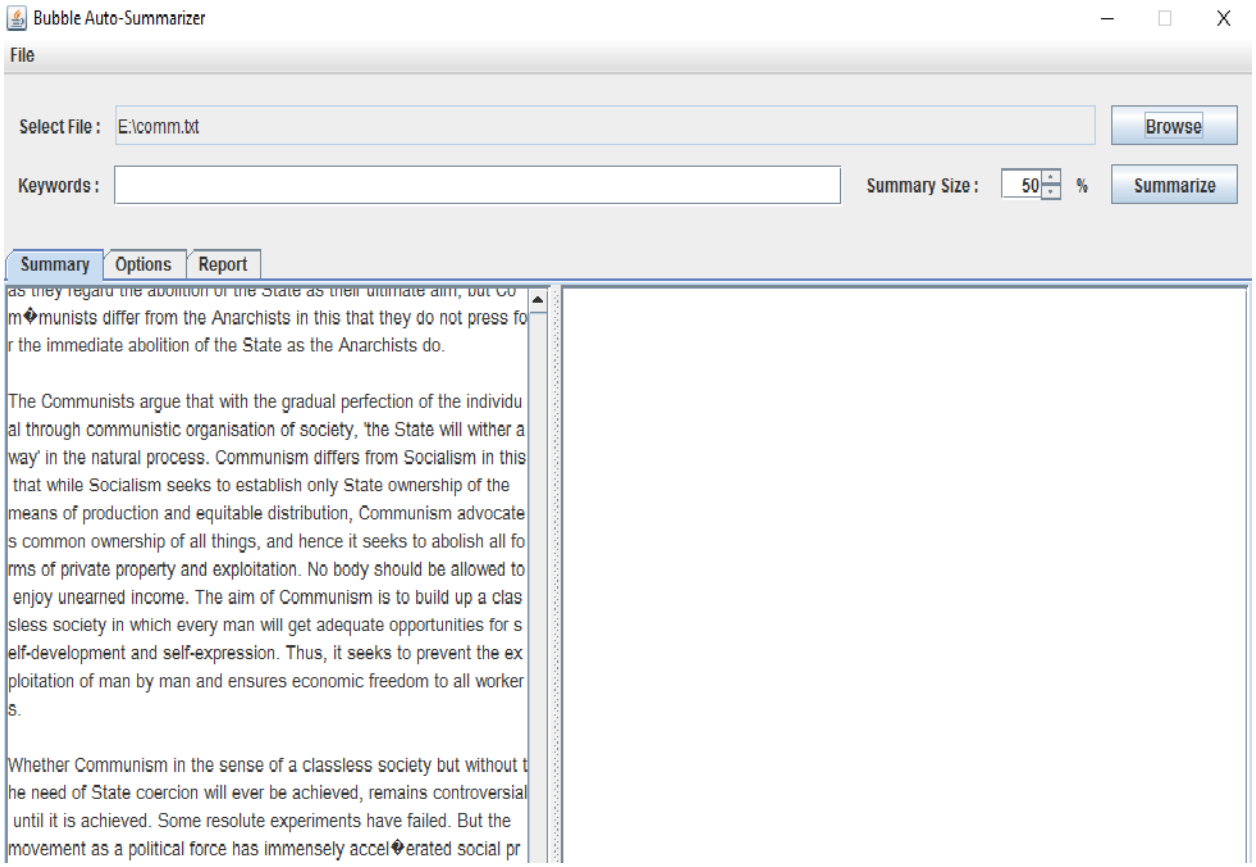
The front end


Bubble Auto-Summarizer
File
Select File :
Browse
Keywords :
Summary Size :
50
%
Summarize
Summary
Options
Report
Open
Look In:
eclipse
configuration
dropins
features
p2
plugins
readme
File Name:
Files of Type:
Plain Text files
Open
Cancel

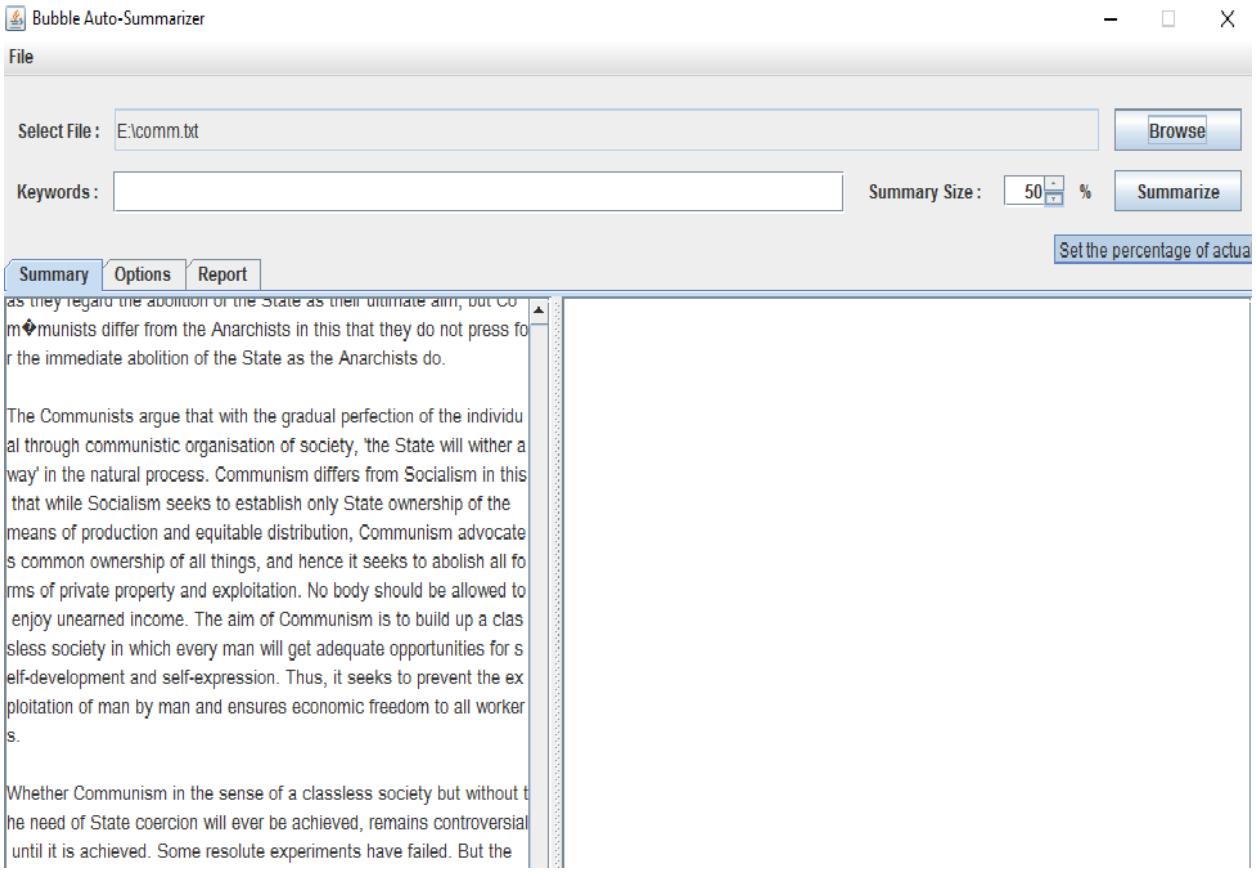

Bubble Auto-Summarizer
File
Select File :
Browse
Keywords :
Summary Size :
50
%
Summarize
Summary
Options
Report
Open
Look In:
software (E:)
DVD RW Drive (D:)
software (E:)
entertainment (F:)
Network
Libraries
Ruchita Chandel
c_mcq
Major project
File Name:
Files of Type:
Plain Text files
Open
Cancel

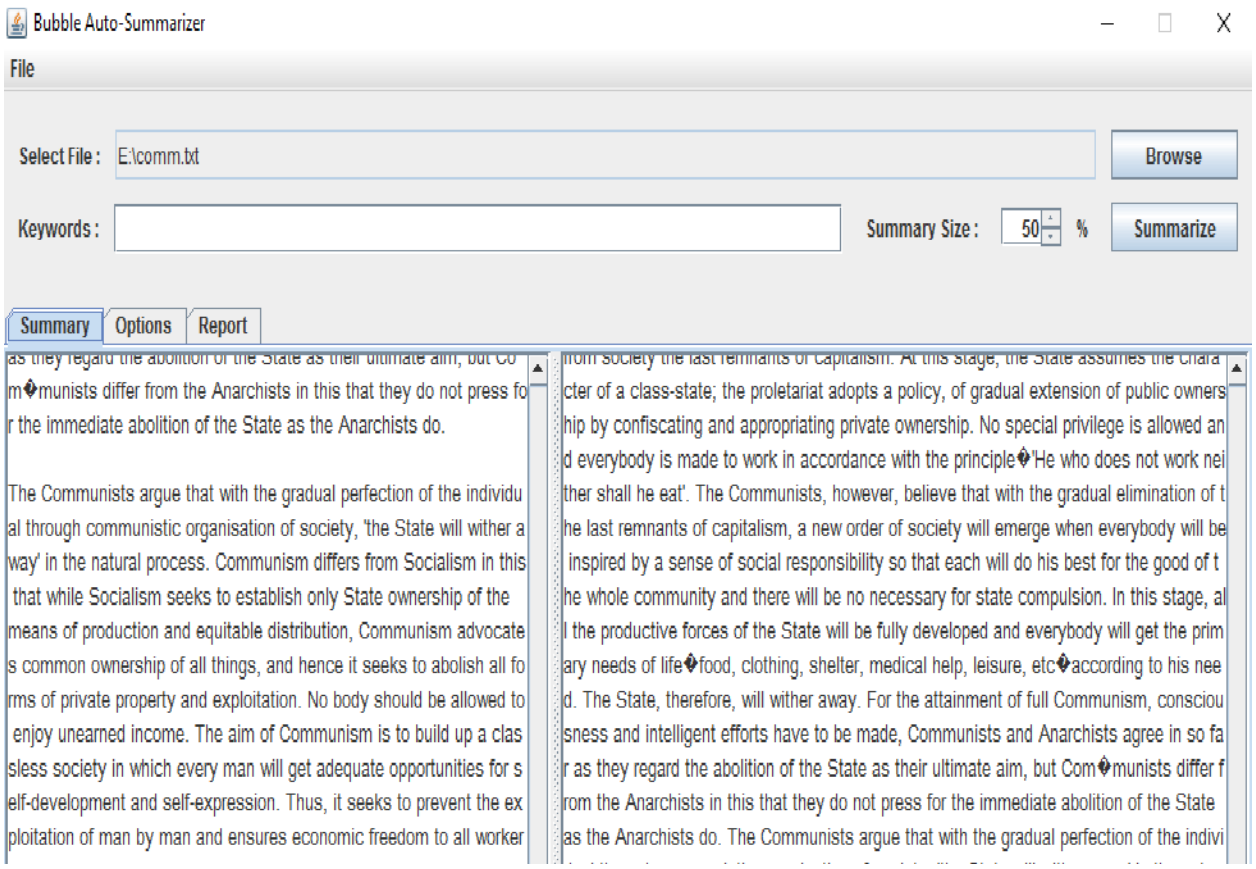


Selection of the file

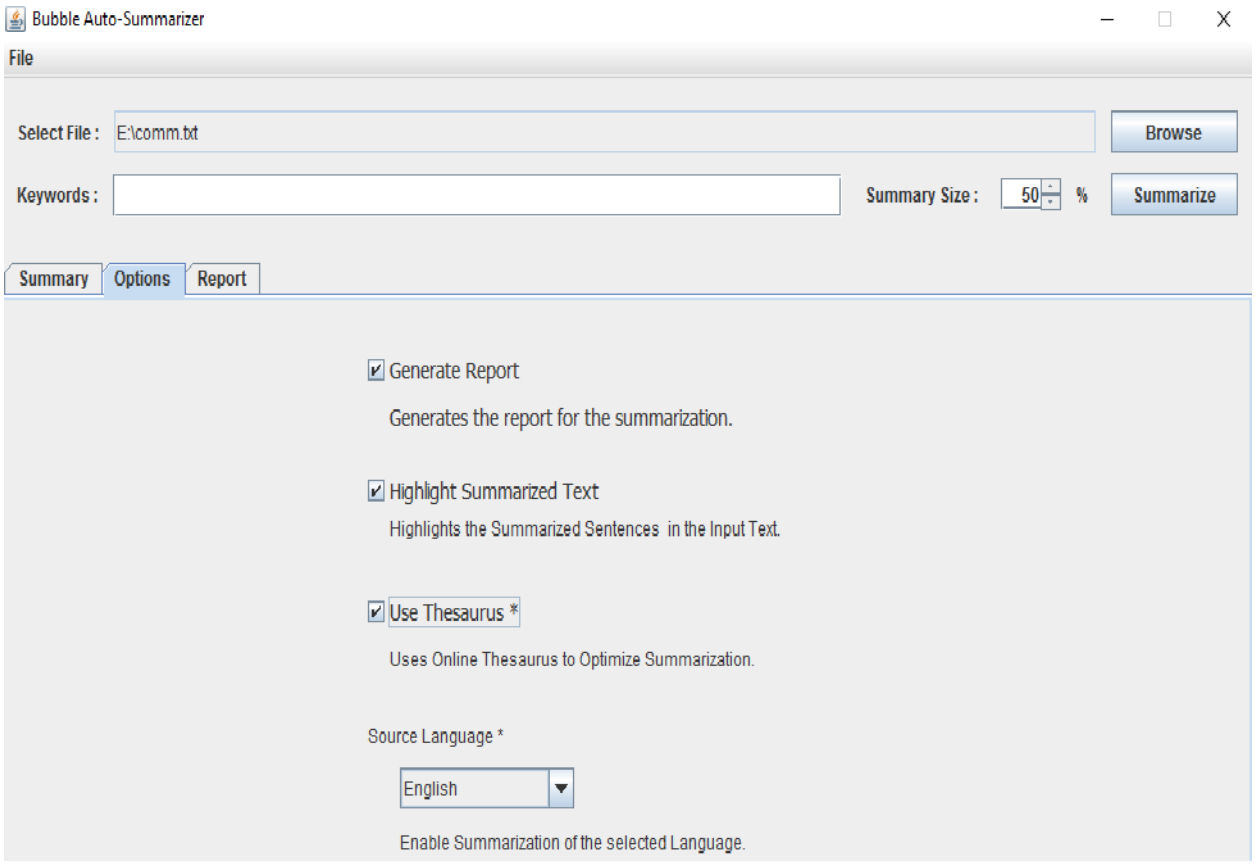


Text input given

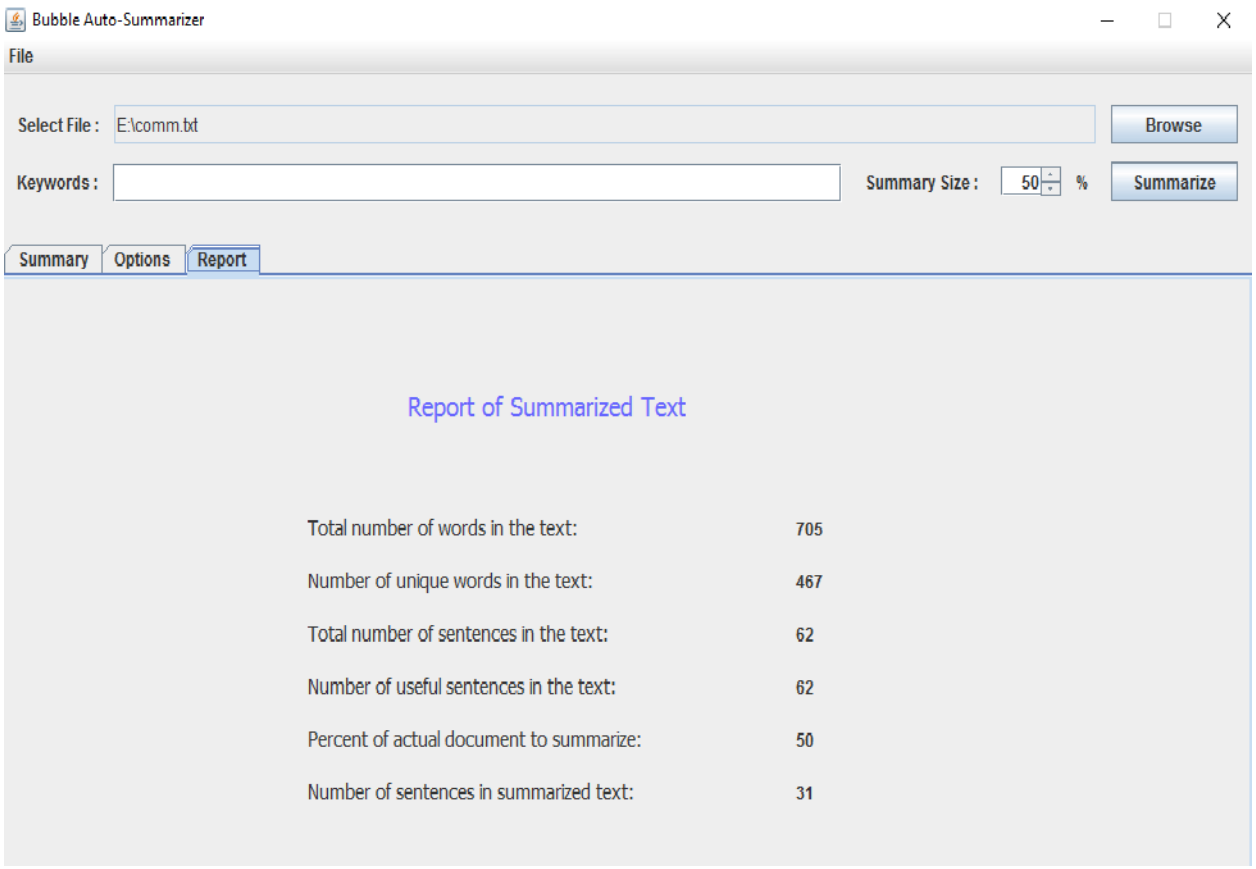

Bubble Auto-Summarizer
File
Select File :
E:\comm.txt
Browse
Keywords :
Summary Size :
50
%
Summarize
Set the percentage of actua
Summary
Options
Report
as they regard the abolition of the State as their ultimate aim, but Co
m munists differ from the Anarchists in this that they do not press fo
r the immediate abolition of the State as the Anarchists do.
The Communists argue that with the gradual perfection of the individu
al through communistic organisation of society, 'the State will wither a
way' in the natural process. Communism differs from Socialism in this
that while Socialism seeks to establish only State ownership of the
means of production and equitable distribution, Communism advocate
s common ownership of all things, and hence it seeks to abolish all fo
rms of private property and exploitation. No body should be allowed to
enjoy unearned income. The aim of Communism is to build up a clas
sless society in which every man will get adequate opportunities for s
elf-development and self-expression. Thus, it seeks to prevent the ex
ploitation of man by man and ensures economic freedom to all worker
s.
Whether Communism in the sense of a classless society but without t
he need of State coercion will ever be achieved, remains controversial
until it is achieved. Some resolute experiments have failed. But the



Summarised in accordance with the ratio



The features asserted



Report of the summarized text

CHAPTER 9

CONCLUSION

The project titled “Automatic Text Summarisation” with the purpose to provide the facility to a user to give a document to the tool to generate a summary of a document so that the user may save his time by not reading the entire large document but picking up the concept described in the document through the generated summary. Utmost care has been taken to see that the results obtained are true and the underlying procedures are implemented in true form.. The project has all the functionalities implemented to work on practical problems.

The tool works by taking a single document from user, extracting the important sentences of document and providing a summary to user. In the proposed project, the generated summary is made more readable by replacing the words in the summary with their synonyms. The system replaces those words in the summary with their meanings which are also present in dictionary. Dictionary is maintained by the system.

9.1 Limitations of the project

-Standalone application.

-Not taking text from any other extension like pdf or from the network.

9.2 Further work

1. Building a single system which can automatically identify the type of input text and use the best method out of these three technique to generate summary.
2. Improving accuracy of features like name entity identifier, location finder etc.
3. Extending the domains for Word2Vec summarizer by training it using various other datasets in diverse domains.

REFERENCES

- G. Sizov, “Extraction-based automatic summarization: Theoretical and empirical investigation of summarization techniques,” 2010.
- M. A. Fattah and F. Ren, “Automatic text summarization,” World Academy of Science, Engineering and Technology, vol. 37, p. 2008, 2008.
- “About wordnet - wordnet - about wordnet,” https://wordnet.princeton.edu/, accessed: 2015-05-15.
- “word2vec - tool for computing continuous distributed representations of words. google project hosting,” https://code.google.com/p/word2vec/, accessed: 2015-05-15.
- J.Ramos,“Usingtf-idftodeterminewordrelevanceindocumentqueries,”inProceedings of the first instructional conference on machine learning, 2003.
- W. T. Chuang and J. Yang, “Extracting sentence segments for text summarization: a machine learning approach,” in Proceedings of the 23rd annual internationalACMSIGIRconferenceonResearchanddevelopmentininformationretrieval. ACM,2000, pp. 152–159.
- S. Ryang and T. Abekawa, “Framework of automatic text summarization using reinforcement learning,” in Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. Association for Computational Linguistics, 2012, pp. 256–265.
- D. Hingu, D. Shah, and S. S. Udmale, “Automatic text summarization of wikipedia articles,” in Communication, Information & Computing Technology (ICCICT), 2015 International Conference on. IEEE, 2015, pp. 1–4.
- H. P. Edmundson, “New methods in automatic extracting,” Journal of the ACM (JACM), vol. 16, no. 2, pp. 264–285, 1969.
- “Wikipedia - wikipedia, the free encyclopedia,” http://en.wikipedia.org/wiki/Wikipedia, accessed: 2015-05-15.

APPENDIX A

Text summarization is a way to condense the large amount of information into a concise form by the process of selection of important information and discarding unimportant and redundant information. With the amount of textual information present in the World Wide Web (www), the area of Automatic Text Summarization (ATS) is becoming very important in the field of information retrieval. The process of condensing a source text in to a shorter version preserving its information content is called summarization. Automated summarization tools can help people to grasp main concepts of information sources in a short time.

Text summarization approaches can be broadly divided into two groups: extractive summarization and abstractive summarization. Extractive summarization extracts salient sentences or phrases from the source documents and group them to produce a summary without changing the source text. Usually, sentences are in the same order as in the original document text. However, abstractive summarization consists of understanding the source text by using linguistic method to interpret and examine the text. The abstractive summarization aims to produce a generalized summary, conveying in information in a concise way, and usually requires advanced language generation and compression techniques[1]. .Early work in summarization started with single document summarization. Single document summarization produces summary of one document. As research proceeded, and due to large amount of information on web, multi document summarization emerged. Multi document summarization produces summaries from many source documents on the same topic or same event. The automatic summarization of text is a well- known task in the field of natural language

processing (NLP). Significant achievements in text summarization have been obtained using sentence extraction and statistical analysis. True abstractive summarization is a dream of researchers [2]. Abstractive methods need a deeper analysis of the text. These methods have the ability to generate new sentences, which improves the focus of a summary, reduce its redundancy and keeps a good compression rate .

One of examples of document summarization is in the field of legal area. The legal experts perform difficult and responsible work. Their resources are sparse and expensive both in time and expertise levels. Thus, the system for concise summarization is necessary in order for experts to be able to effectively and in short time find compressed and restated content of relevant judicial documents, including laws and their proposals, relevant court decisions or tribunal process summarizations etc. In the medical branch,

there is often overload of information and it is requirement in many cases for the medical personal to find relevant information about patient's conditions every time. This involves crawling of many documents and patient's record in order to gain necessary information. In this area the text summarization specifically adjusted to medical domain is of considerable use, saving time resources and optimizing availability of medical experts.

On the internet, there are many examples of the summarizations used. For instance, news portals like Microsoft News2,Google1 or Columbia Newsblaster3 are relying of such techniques in order to provide short news summaries to their visitors. There are also service providing blog summarization and aggregation4 and opinion survey systems.

Further, there is also application of document summarization for PDA devices with small screen, where the only limited screen size and time are available for users to read. For the businesses it is also important to have available summarizations of meetings coupled possibly with speech recognition systems, to provide "meeting minutes" in short time and without using excessive human and other resources. For the handicapped people, the text summarization systems are also of great help. They can save much time for readers of such documents using speech synthesis technologies, and in order to be able to recognize and separate important and less important content according to their interests.

APPENDIX B

FORMAT OF CD CONTAINING COMPUTER SOFTWARE

This CD contains all the information related to how to run the software. It also includes requirements of the software which includes minimum hardware requirement of the system, platform required; soft wares need to be installed in the system, version of the software used and steps of running of the .exe file of the software.

B.1 Installing JDK /JRE

1. Double-click the downloaded disk image (DMG) file. Follow the screen instruction to install JDK/JRE

2. Eject the DMG file .

3. To verify your installation , open a "Terminal" and issue these commands.

B.2 Installing Eclipse System

1. Unzip file "eclipse-jee-juno-SR2-win32zip"

2. To add eclipse to the "Start- Menu" and customize its properties , right click on "eclipse.exe" and select "Create Shortcut" from the menu.

3.Right click on the shortcut file that's created and select "Rename" from the Menu and enter "Eclispe".

B.3 Integrating JDK into Eclipse

1. Right click on the project>Properties

2. Select "JavaBuildPath" on left , "JRESystemLibrary" and click Edit .....

3. Select "WorkSpaceDefaultJRE".

4.Click "InstalledJREs"

5. If you see JRE you want in the list select it .

B.4 Content of CD

a) JDK (Java Development Kit)

b) Documentation (Project Report)

c) Eclipse Juno