Improved Stemming approach used for Text Processing in Information Retrieval System

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By
Deepika Sharma
801032006

Under the supervision of:
Dr. Deepak Garg
Associate Professor, CSED

Mr. V. K. Bhalla
Assistant Professor, CSED

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

May 2012

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, **"Improved Stemming Approach Used for Text Processing in Information Retrieval System",** in partial fulfilment of the requirements for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Deepak Garg and Mr. V. K. Bhalla and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Deepika Sharma.

Signature:

(Deepika Sharma)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Deepak Garg.

(Dr. Deepak Garg)
CSE Dept.
Thapar University,
Patiala

V.K.Bhalla

(Mr. V. K. Bhalla)
CSE Dept.
Thapar University,
Patiala

Countersigned By:

(Dr. Maninder Singh)
Head
CSE Dept.
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

ACKNOWLEDGEMENT

First of all, I am thankful to God for his blessings and showing me the right direction. With His mercy, it has been made possible for me to reach so far.

It gives me great pleasure to express my gratitude towards the guidance and help I have received from Dr. Deepak Garg. I am thankful for his continual support, encouragement, and invaluable suggestion. He not only provided me help whenever needed, but also the resources required to complete this thesis report on time.

I extend my thanks to Mr. V. K. Bhalla for sharing his expertise and time to help me accomplish this work.

I am also thankful to Dr. Maninder Singh, Head, Computer Science and Engineering Department for his kind help and cooperation. I express my gratitude to all the staff members of Computer Science and Engineering Department for providing me all the facilities required for the completion of my thesis work.

I would like to say thanks to all my friends. I want to express my appreciation to every person who contributed with either inspirational or actual work to this thesis.

Last but not the least I am highly grateful to all my family members for their inspiration and ever encouraging moral support, which enables me to purse my studies.

Deepika Sharma.

Deepika Sharma

ABSTRACT

Nowadays, Internet is one of the main information providers for millions of people. There are about trillions of pages practically about all matters available on the Web. One can find information related to any topic from the Web, however with this much vast resource of information at hand there comes certain challenges. Some of these challenges are how to retrieve the relevant information from such huge collection of documents and how to make this retrieval more efficient. There are number of solutions to solve these problems like the use of classical information retrieval systems (library system) or web based information retrieval system (search engines). These information retrieval systems perform lots of steps before retrieving the information. One such step is stemming which is one of the important processes in text processing operations. Stemming is used to map the morphologically related words to a common stem or root word, thus reducing the index size and greatly enhancing the recall value.

This report has discussed various information retrieval methods and ways to improve the retrieval results using stemming approach. This report proposes few improvements on the recently used stemmers. It mainly focuses on improving the algorithm for finding the valid suffix-pair among various words present in the document.

The result of this algorithm can be used to make classes of the words and thus can be used in creating index tables of relatively lesser index entries. The whole process of finding such valid suffix-pairs can be done in O (n logn) which is better than other approaches used.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Background

Internet is the ultimate source of information in the present world. One can find millions of documents related to any topic on the web. To have access to all the relevant documents related to a user query one need to have a system which can efficiently and effectively retrieve them. Information retrieval systems are widely used to help users find the required documents as per their needs. However, these information retrieval systems face a lot of challenges during the retrieval of information like

- How to maximize the retrieval effectiveness
- How to keep a check on duplicate documents
- How to remove irrelevant documents from retrieved results
- How to optimize the retrieved results by placing highly ranked documents above the less ranked documents.

Information retrieval systems can be broadly categorized into two categories mainly based on the amount of corpus they work upon i.e. Classical Information Retrieval Systems and Web Based Information Retrieval Systems [1].

This chapter discusses the following points:

- Introduction to Classical Information Retrieval Systems and Web Based Information Retrieval Systems i.e. Search Engines.
- Steps that are being followed by these systems while information retrieval process.
- Various measures to make the retrieval more efficient.

1.2 Classical Information Retrieval

In the field of information retrieval the following classic problem setting is studied: A user tries to satisfy an information need in a given collection of documents. For that purpose the user inputs a request into the information retrieval system containing the collection. The goal of the system is to retrieve documents with information content that is relevant to the user's information need.

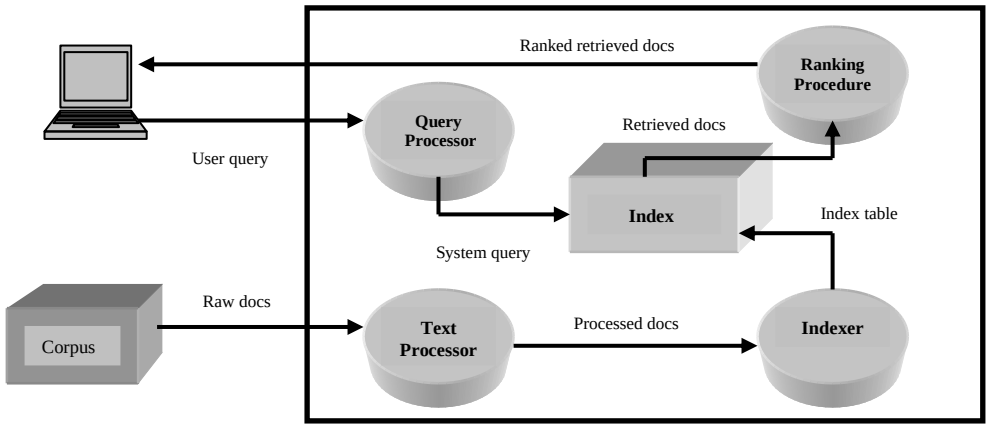

Figure 1.1 Information Retrieval System

An information retrieval system has to handle two tasks:

- Processing the collection to build an internal data structure that allows efficient access to the relevant document.
- Processing queries to convert them to systems understandable form.

There are various strategies, called models to determine which documents to return. The basic model simply returns all documents that contain at least one of the query terms or all of the query terms. The logic model extends the basic model with the logical operators AND, OR and NOT.

1.3 Types of Information Retrieval Models

Information retrieval models are the mathematical models which provide a framework for defining the search process. These retrieval models [2] make various assumptions about the relevance of the retrieved result to simplify problem:

- Topical versus user relevance
- Binary versus multi-valued relevance

Information retrieval models can be broadly classified into three main types [2]:

- Boolean retrieval model
- Vector space model
- Probabilistic model

Both vector space model and probabilistic model come under the category of statistical models because they use the statistical information to determine the relevance of documents. This information is in the form of term frequencies with respect to a query. Boolean model is based on the exact match principle whereas the statistical models are based on the best match principle.

1.3.1 Boolean Retrieval Model

In this model there are two possible outcomes after query processing i.e. either TRUE or FALSE. It is based on the exact matching retrieval process i.e. the model will retrieve the results only if there are exact match between the query text and the stored document text. This is one of the oldest and primitive types of information retrieval model which uses simplest form of ranking of the documents. Queries are usually specified using Boolean operators like AND, OR and NOT.

For example: A user wants to search for news articles to "Obama". Then besides entering "Obama" in his query he must specify other necessary details using Boolean operators to get the relevant documents. Following are the few instances of query that he may enter to make the retrieval more relevant:

- President AND Obama
- President AND Obama AND NOT (university OR college)
- President AND Obama AND biography AND life AND birthplace AND NOT (university OR college)
- President AND Obama AND (biography OR life OR birthplace) AND NOT (university OR college)

Advantages

1. Results are predictable and relatively easy to explain.
2. Many different features can be incorporated within a single query.

3. Efficient processing can be achieved since many documents can be eliminated from search.

Disadvantages

1. Effectiveness of the retrieval results depends entirely on the type of user query. Efficient use of boolean operators is required to make good queries.
2. Simple queries like one word query without the use of logical operators usually do not retrieve satisfactory results.
3. It is difficult to make complex query for every possible case.

1.3.2 Vector Space Model

In this model both documents and query are represented by a vector of terms weights. Term weight is the numerical value assigned to each term present either in the document or in the query representing the frequency of the occurrence of that term. From each document a collection of terms is made and corresponding to it a matrix is drawn where the values present in the matrix represent the weight of a particular term in a particular document.

Consider a collection of documents $Doc_1, Doc_2, ..., Doc_n$ and each document having the collection of terms as $Doc_i = (d_{i1}, d_{i2}, ... , d_{it})$, then the above collection can be represented in a matrix form as follows:

	Term 1	**Term 2**	**…**	**Term t**
Doc1	d_{11}	d_{12}	…	d_{1t}
Doc2	d_{21}	d_{22}	…	d_{2t}
:	:	:	:	:
Docn	d_{n1}	d_{n2}	…	d_{nt}

Figure 1.2 Weight Matrix of Vector Space Model

Consider the following example of four documents

- D1= Cook Indian and Continental Food
- D2 = South Indian Food and Deserts
- D3 = Indian Recipes Home page-Indian Cuisines

- D4 = How to cook South Indian Food and Punjabi Food

TERMS	**DOCUMENTS**			
	D1	**D2**	**D3**	**D4**
Indian	1	1	2	1
Recipes	0	0	1	0
Deserts	0	1	0	0
Food	1	1	0	2
Continental	1	0	0	0
Cook	1	0	0	1
Homepage	0	0	1	0
And	1	1	0	1
South	0	1	0	1
Punjabi	0	0	0	1
Cuisines	0	0	1	0

Figure 1.3 Example of Matrix Construction in Vector Space Model

Similar matrix can be drawn for the terms present in the user query. For example user submits a query on “Indian Food” then corresponding to this query Doc 3 and Doc 4 are ranked high as compared to Doc1 and Doc2. This result can be easily visualised using a 3-d picture as shown below:

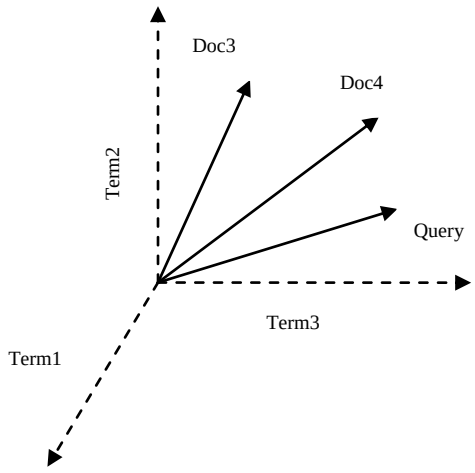

Figure 1.4 3-D view of Results in Vector Space Model

Documents are ranked by calculating the similarity and dissimilarity measure among the query and document terms.

Advantages

1. Simple computational framework for ranking.
2. Any similarity measure or term weighting scheme could be used.

Disadvantages

1. Assumption of term independence.
2. No predictions about techniques for effective ranking.

1.3.3 Probabilistic Model

This 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 [3]. Human's uncertain needs and ever changing requirements are considered as the base of probability ranking principle. User may ask queries differently and same information can be represented using different words and formats. So, this principle takes into account all these variations and gives the results based on the statistical analysis of the distribution of words.

Advantages

1. They provide users with a relevance ranking of the retrieved documents. Thus the output of the retrieval is controlled by setting a relevance threshold.
2. It is easier to formulate queries because there is no need to learn any query language.

Disadvantages

1. They have limited expressive power for some of the operations. For example, the NOT operation can not be represented because only positive weights are used.
2. There is no well defined structure to express some of the important linguistic features like phrases, same information being used in different tense forms.

3. The computation of the relevance scores can be computationally expensive.
4. The result of the retrieval is displayed in the form of their ranking based on the relevance but with a limited view of the information needed and there is no suggestion on how to improve the queries.
5. To improve the retrieval results one has to submit larger and complex queries.

Table 1.1 Comparison of IR Models [4]

	Boolean Model	**Vector Space Model**	**Probabilistic Model**
Goal	• Capture conceptual structure and contextual information	• Rank the output based on similarity	• Probability of relevance
Methods	• Boolean operators are used: AND, OR, NOT	• Cosine Measure	• Use of different models
Advantage	• Easy to implement • Computationally Efficient • Expressiveness and Clarity	• Retrieved set is ranked as per relevance • Always gives the best match results	• Query terms are ranked based on their probability of relevance • User can control the output by setting a relevance threshold.
Issues	• Knowledge specific to the use of Boolean operators is required • Difficult to control output • No ranking is done • No weighting of index or query terms are done • No uncertainty measures are calculated	• NOT operator can't be expressed • Limited expressive power • Computationally intensive • Assumes that the terms are independent	• Estimation of needed probability • Prior language knowledge is needed • Assume terms independence • Lack of structure to visualize the retrieved set

1.4 Information Retrieval on the Web

Information Retrieval on the Web is a variant of classical information retrieval [1]. As in classical information retrieval, a user tries to satisfy an information need in a collection of documents. In this case the collection of documents consists of all the web pages in the publicly accessible web. Given a user query the goal is to retrieve high quality web pages that are relevant to the user's need. So, finding high quality documents is an additional requirement that arises in the web context.

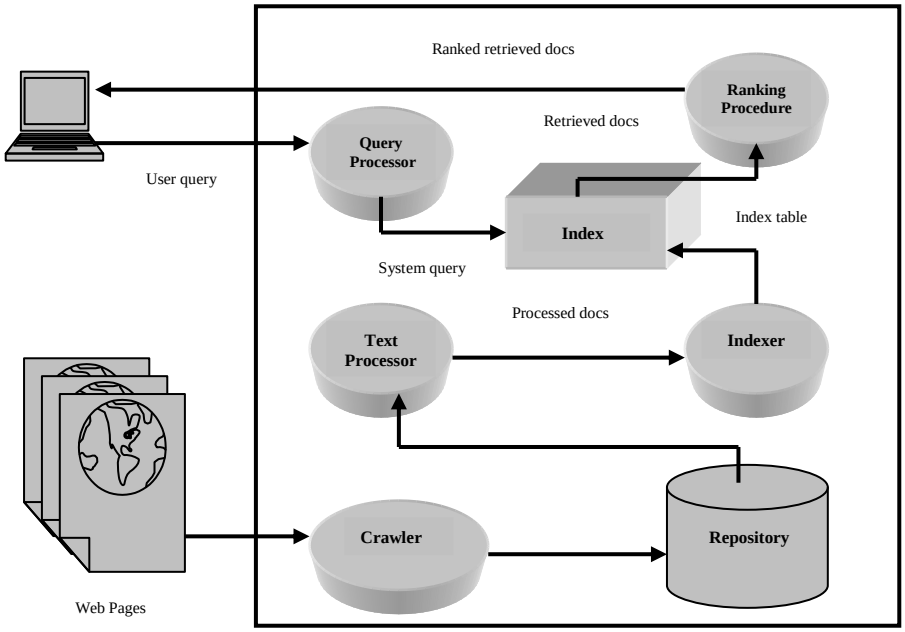

Figure 1.5 General Purpose Search Engine

General purpose search engines are used to index a sizeable portion of the web across all topics and domains to retrieve the information. Each such Engine consists of three major components:

- A spider or crawler [5] browses the web by starting with a list of URLs called the seeds. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs which are visited recursively to form a huge collection of documents called corpus. The corpus is typically augmented with pages obtained from direct submissions to search engines and various other sources. Each crawler has different policies with respect to

which links are followed, how deep various sites are explored, etc. As a result, there is surprisingly little correlation among corpora of various engines [6].

- The indexer processes the data and represents it usually in the form of fully inverted files. However, each major Search Engine uses different representation schemes and has different policies with respect to which words are indexed.
- The query processor which processes the input query and returns matching answers, in an order determined by a ranking algorithm. It consists of a front end that transforms the input and brings it to a standard format and a back end that finds the matching documents and ranks them.

1.5 IR Tools on the Web

Information from web can be retrieved by number of tools available ranging from general purpose search engines to specialized search engines. Following are the most commonly used web IR tools [1]:

1. **General-Purpose Search Engine**: They are the most commonly used tool for information retrieval. Google, AltaVista, Excite are some of the examples. Each of them has its own set of web pages which they search to answer a query.

2. **Hierarchical Directories**: In this approach the user is required to choose one of a given set of categories at each level to get to the next level. For example, Yahoo! or the dmoz open directory project.

3. **Specialized Search Engines**: These search engines are specialized on an area and provides huge collection of documents related to that specific area. For e.g. PubMed, a search engine specialized on medical publications. [1]

4. **Other Search Paradigms**: There are various other search paradigms. A Search-by-Example feature exists in various incarnations. Also various collaborative filtering approaches and notification systems exist on the Web.[1]

1.6 Comparison with Classical Information Retrieval

Basically the differences between classical information retrieval system and web based information retrieval system can be partitioned into two parts, namely differences in the documents and differences in the users [1].

Differences in the documents:

- **Hypertext**: Documents present on the web are different from general text-only documents because of the presence of hyperlinks. It is estimated that there are roughly 10 hyperlinks present per document.
- **Heterogeneity of document**: The contents present on a web page are heterogeneous in nature i.e., in addition to text they might contain other multimedia contents like audio, video and images.
- **Duplication**: On the web, over 20% of the documents present are either near or exact duplicates of other documents and this estimation has not included the semantic duplicates yet.
- **Number of documents**: The size of web has grown exponentially over the past few years. The collection of documents is over trillions and this collection is much larger than any collection of documents processed by an information retrieval system. According to estimation, web currently grows by 10% per month.
- **Lack of stability**: Web pages lack stability in the sense that the contents of Web pages are modified frequently. Moreover any person using internet can create a web pages even if it contains authentic information or not.

The users on the web behave differently than the users of the classical information retrieval systems. The users of the latter are mostly trained librarians whereas the range of Web users varies from a layman to a technically sound person. Typical user behaviour shows:

- **Poor queries**: Most of the queries submitted by users are usually short and lack useful keywords that may help in the retrieval of relevant information.
- **Reaction to results**: Usually users don't evaluate all the result screens, they restrict to only results displayed in the first result screen.
- **Heterogeneity of users**: There is a wide variance in web users and their web experience.

Thus, the main challenge of information retrieval on the web is how to meet the user needs given the heterogeneity of the web pages and the poorly made queries.

To meet theses challenges, any type of information retrieval system either a classical one or web based has to undergo four essential steps:

- **Document processing**: The text present in the corpus is processed into a predefined format; stems of the words are extracted i.e. words in the document are collated to their root words to make index entries.
- **Query processing**: User queries are tokenized into understandable segments, these segments are parsed and a general query representation is made which is then used for matching query terms with the inverted index entries.
- **A search and matching function**: Each document in the corpus is searched for the query terms and based on the matching of terms; each document is given some matching score. However, different systems may adopt different models for performing this searching and matching.
- **A ranking capability**: Based on the similarity score of each document they are retrieved as a result of user query in the decreasing order of their relevance i.e. the documents with higher relevance will be displayed first than others with lesser relevance.

Lots of work has been done to improve these basic modules to make retrieval more efficient, however, this report will discuss one very basic and important step in document and query processing i.e. Stemming.

1.7 Text Processing

The most common feature of a document or a query is that they both consist of collection of terms. These terms are represented by some structure like there are set of grammatical rules, preposition and blank spaces in between. So, the first and the foremost step in any information retrieval system is the text processing [7] of the terms present in the collection of documents, the whole process of text processing can be visualized by figure 1.6.

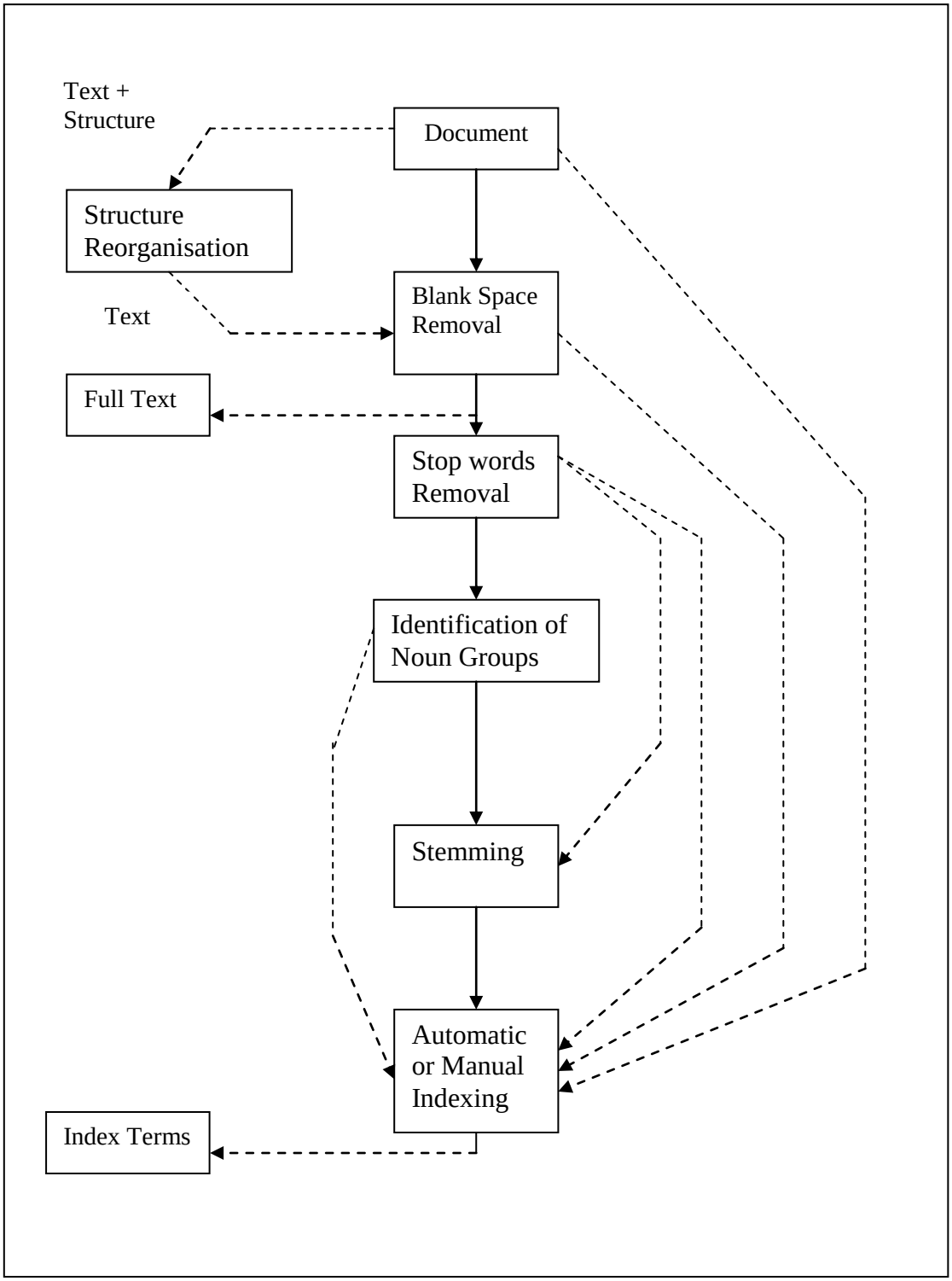

Figure 1.6 Text Processing Operations [7]

Every information retrieval system adopts a full text logical view (or representation) of the documents, they reduce the document into a set of representative keywords. This process can be accomplished by the removal of blank spaces present in between the terms, and then elimination of stop words such as articles and connectives. Once

the set of terms are obtained they are identified for the presence of noun groups and eliminate adjectives, adverbs, and verbs. Then there is use of stemming approach which reduces distinct words to their common grammatical root thereby creating the distinct index terms. This complete process is called as text processing.

These text operations reduce the complexity of the document representation and allow moving the logical view from that of a full text to a set of index terms. A full text is clearly the most complete logical view of a document but its usage implies higher computational costs. A small set of categories provides the easiest logical view of a document but its usage might lead to retrieval of poor quality. Several intermediate logical views of a document might be adopted by an information retrieval system as described in figure 1.6. Besides adopting any of the intermediate representation, the retrieval system might also recognize the internal structure present in the document like its chapters, sections, subsection etc. This information on the structure of the document might be useful and is required by the text retrieval models as discussed in section 1.3.

This thesis will discuss in details one of the most researched [12] [13] [20] [23] [24] and important text processing operation called as stemming.

1.8 Stemming

Stemming is a process to convert the words having morphological similarity into one common form. For example, words like prepare, preparation, preparing are all derived from one common root word “prepar”. So, if a user enters a query related to “How to prepare food?” and the stored documents in the corpus have topics like “Food Preparation”, “Steps of preparing food” then he may miss out all these related documents if the stemming is not used. However, with the use of stemming words like preparation, preparing, prepare will be stemmed down to their root word prepar and would yield all the relevant documents related to the query. Moreover, while constructing the index table the numbers of entries are also reduced because instead of storing all the words separately only root words are stored in the index table.

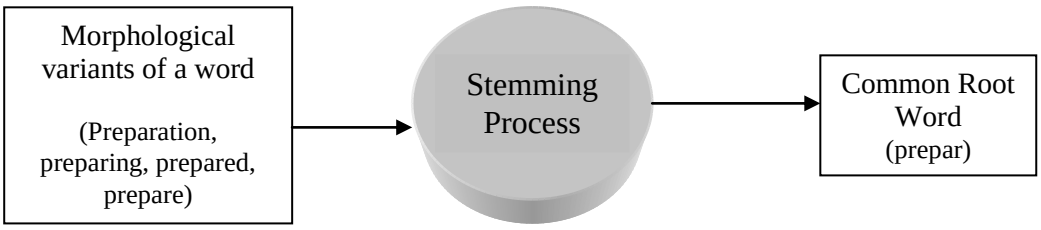

Figure 1.7 The Stemming Process

Thus stemming provides two basic advantages;

- It is used to increase the Recall rate of the information retrieval. Recall rate is defined as the number of relevant documents retrieved by total number of documents retrieved.
- It helps to save the memory by reducing the entries in the index table, thereby reducing the size of index table.

1.9 Structure of the Thesis

The rest of thesis is organized in the following order:

Chapter-2: This chapter will provide the overview of all recent work done in the area of stemming. It starts with the introduction to stemming, describes various conflation methods used to conflate a word to its derivational stem or root word. Then discusses two main types of stemming approaches namely rule based approach and statistical approach. Then there will a comparative analysis of these approaches on two main parameters like stemmer strength and computation cost.

Chapter-3: This chapter gives the problem statement and methodology used to solve the problem. There will be a gap analysis of the work done on stemming till date and a new approach is proposed to fill these gaps or as one more alternative to be used for stemming.

Chapter-4: This chapter provides the solution to the problem discussed in chapter-3. Complete discussion of the data structure as well as algorithm used will be done. Also it explains the solution. There is a comparative analysis of the techniques done and complete complexity analysis of the proposed algorithm.

Chapter-5: This chapter gives the conclusion of the thesis with the future scope of the topic.

CHAPTER 2
LITERATURE SURVEY

With the enormous amount of data available online, it is very essential to retrieve accurate data for some user query. There are lots of approaches used to increase the effectiveness of online data retrieval. The traditional approach used to retrieve data for some user query is to search the documents present in the corpus word by word for the given query. This approach is very time consuming and it may miss some of the related documents of equal importance. Thus to avoid these situations, stemming has been extensively used in various information retrieval systems to increase the retrieval accuracy.

Stemming is the conflation of the variant forms of a word into a single representation, i.e. the stem. For example, the terms presentation, presenting, and presented could all be stemmed to present. The stem does not need to be a valid word, but it must capture the meaning of the word. In information retrieval systems stemming is used to conflate a word to its various forms to avoid mismatches between the query being asked by the user and the words present in the documents. For example, if a user wants to search for a document on “How to cook” and submits a query on “cooking” he may not get all the relevant results. However, if the query is stemmed, so that “cooking” becomes “cook”, then retrieval will be successful.

Stemming has been extensively used to increase the performance of information retrieval systems. For some international languages like Hebrew, Portuguese, Hungarian [9], Czech, and French and for many Indian languages like Bengali, Marathi, and Hindi [8] stemming increase the number of documents retrieved by between 10 and 50 times [10]. For English though the results are less dramatic but better than the baseline approach where no stemming is used. Stemming is also used to reduce the size of index files. Since a single stem typically corresponds to several full terms, by storing stems instead of terms, compression factor of 50 percent can be achieved.

The terms in a document can be stemmed before indexing time or before search time. The direct advantage of stemming at the time of indexing is that indexing will be done efficiently and also index file will be in compressed form. As index terms are already stemmed, this operation requires no resources at search time, and the index file will be compressed as described above. The main disadvantage of performing stemming at index time is that the information about the full term will be lost like the form of verb being used; grammatical usage etc, additional storage to store both the stemmed and unstemmed words is also one of the disadvantages.

2.1 Use of Stemmer in Searching

To see how a stemmer can be used in searching, consider the following example from the CATALOG system [11]. In this system, the terms in the user query or present in the stored documents are stemmed during search time instead of indexing time. It gives the users a facility to type their queries at the prompt with the string"Look for". For example:

Look for: Active users

As soon as the system gets this request it attempts to find all the documents about active users. In order to accomplish this matching process CATALOG takes each term in the user query and will try to match those terms with the terms present in the database having same stem or root word. If any possibly related terms are found, CATALOG presents them to the user for selection. In the case of the query term "users," for example, CATALOG might respond as follows:

Search Term: users

	Term	Occurrences
1.	user	15
2.	users	1
3.	used	3
4.	using	2

Which terms (0 = none, CR = all):

The user can select the result based on their occurrence in the document.

The above discussed method used for stemming provides a naïve user to have access to all the conflated terms of his requirement without requiring any system knowledge or of searching techniques. It also allows experienced searchers to focus their attention on other search problems. Stemming may not be always appropriate so the user may turn off the stemmer and select only some of the results of his interest to reduce the chances of false matches.

2.2 Conflation Methods

For achieving stemming one needs to conflate a word to its various variants. Figure 2.1 shows a various conflation methods that can be used in stemming. Conflation of words or so called stemming can either be done manually by using some kind of regular expressions or automatically using stemmers. There are four automatic approaches namely Affix Removal Method, Successor Variety Method, n-gram Method and Table lookup method [11].

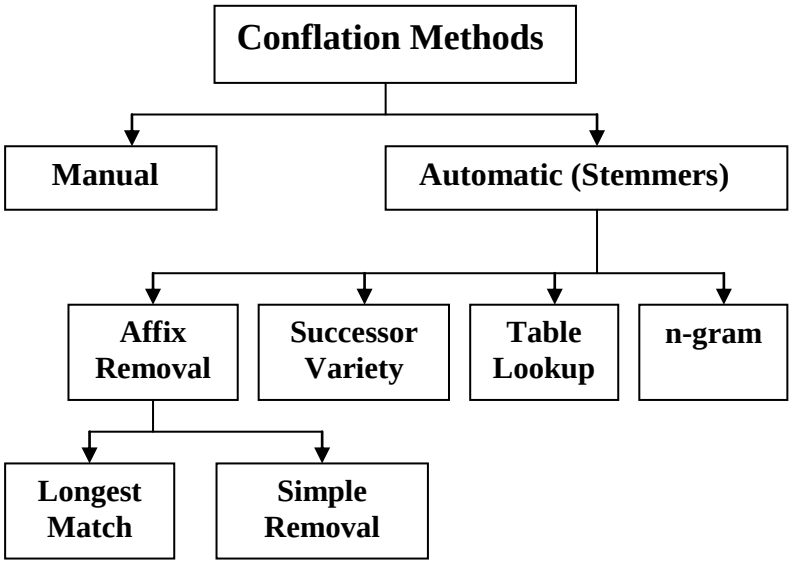

Figure 2.1 Conflation Method [11] [31]

There are several criteria for judging stemmers: correctness, retrieval effectiveness, and compression performance. There are two ways stemming can be incorrect--overstemming and understemming. When a term is overstemmed, too much of it is removed. Overstemming can cause unrelated terms to be conflated. The effect on IR performance is retrieval of nonrelevant documents. For example the terms ‘legal’ and

‘legging’ are derived from two unrelated terms but due to over stemming may be stemmed to the term ’leg’ which may yield incorrect results. Understemming is the removal of too little of a term. Understemming will prevent related terms from being conflated. For example the terms ‘absorption’ and ‘absorbing’ are derived from same root word ‘absorb’ but due to understemming they may not be stemmed under same root. The effect of understemming on IR performance is that relevant documents will not be retrieved. Stemmers can also be judged on their retrieval effectiveness--usually measured with recall and precision and on their speed, size, and so on. Finally, they can be rated on their compression performance. Stemmers for IR are not usually judged on the basis of linguistic correctness, though the stems they produce are usually very similar to root morphemes, as described below.

2.2.1 Affix Removal Method

The affix removal method removes suffix or prefix from the words so as to convert them into a common stem form. Most of the stemmers that are currently used use this type of approach for conflation. Affix removal method is based on two principles one is iterations and the other is longest match. A simple example of an affix removal stemmer is one that removes the plurals from terms. A set of rules for such a stemmer is as follows [11].

If a word ends in "ies" but not "eies" or "aies"
Then "ies" -> "y"
If a word ends in "es" but not "aes", "ees", or "oes"
then "es" -> "e"
If a word ends in "s", but not "us" or "ss"
then "s" -> NULL

An iterative stemming algorithm is simply a recursive procedure, as its name implies, which removes strings in each order-class one at a time, starting at the end of a word and working toward its beginning. No more than one match is allowed within a single order-class, by definition. Iteration is usually based on the fact that suffixes are attached to stems in a "certain order, that is, there exist order-classes of suffixes.
The longest-match principle states that within any given class of endings, if more than one ending provides a match, the one which is longest should be removed. The first

stemmer based on this approach is the one developed by [12] Lovins (1968); MF Porter [13] (1980) also used this method. However, Porter's stemmer is more compact and easy to use then Lovins. YASS is another stemmer based on the same approach; it is however language independent is nature.

2.2.2 Successor Variety Method

Successor variety stemmers [14] are based on work in structural linguistics which attempted to determine word and morpheme boundaries based on the distribution of phonemes in a large body of utterances. The stemming method based on this work uses letters in place of phonemes, and a body of text in place of phonemically transcribed utterances.

Hafer and Weiss [14] formally defined the technique as follows:

Let α be a word of length n and α_i, is a length i prefix of α. Let D be the corpus of words. $D\alpha_i$ is defined as the subset of D containing those terms whose first i letters match α_i exactly. The successor variety of α_i, denoted $S\alpha_i$, is then defined as the number of distinct letters that occupy the i + 1st position of words in $D\alpha_i$. A test word of length n has n successor varieties $S\alpha_{i,}$ $S\alpha_{i+1}$,.....$S\alpha_n$.

Successor variety stemmers [11] use the frequencies of letter sequences in a body of text as the basis of stemming. In less formal terms, the successor variety of a string is the number of different characters that follow it in words in some body of text. Consider a body of text consisting of the following words, for example.

back, beach, body, backward, boy

To determine the successor varieties for "battle," for example, the following process would be used. The first letter of battle is "b." "b" is followed in the text body by three characters: "a," "e," and "o." Thus, the successor variety of "b" is three. The next successor variety for battle would be one, since only "c" follows "ba" in the text When this process is carried out using a large body of text, the successor variety of substrings of a term will decrease as more characters are added until a segment

boundary is reached. At this point, the successor variety will sharply increase. This information is used to identify stems.

Once the successor varieties for a given word have been derived, this information must be used to segment the word. Hafer and Weiss [14] discuss four ways of doing this.

1. Using the *cutoff method*, some cutoff value is selected for successor varieties and a boundary is identified whenever the cutoff value is reached. The problem with this method is how to select the cutoff value--if it is too small, incorrect cuts will be made; if too large, correct cuts will be missed.

2. With the *peak and plateau method*, a segment break is made after a character whose successor variety exceeds that of the character immediately preceding it and the character immediately following it. This method removes the need for the cutoff value to be selected.

3. In the *complete word method*, a break is made after a segment if the segment is a complete word in the corpus.

4. The *entropy method* takes advantage of the distribution of successor variety letters. The method works as follows. Let $|D_{\alpha i}|$ be the number of words in a text body beginning with the i length sequence of letters α. Let $|D_{\alpha ij}|$ be the number of words in $D_{\alpha i}$ with the successor j. The probability that a member of $D_{\alpha i}$ has the successor j is given by $\frac{|D_{\alpha ij}|}{|D_{\alpha i}|}$. The entropy of $|D_{\alpha i}|$ is

$$H_{\alpha i} = \sum_{p=1}^{26} - \frac{|D_{\alpha ij}|}{|D_{\alpha i}|} \cdot \log_2 \frac{|D_{\alpha ij}|}{|D_{\alpha i}|}$$

Using this equation, a set of entropy measures can be determined for a word. A set of entropy measures for predecessors can also be defined similarly. A cutoff value is selected, and a boundary is identified whenever the cutoff value is reached.

To illustrate the use of successor variety stemming, consider the example below where the task is to determine the stem of the word CAREFUL.

Test Word: CAREFUL

Corpus: CAREFUL, CAR, COOK, CEAT, CARES, CARED, CARING, CARD, CLIP, CARELESS, CAREFREE.

Prefix	Successor Variety	Letters
C	4	A, O, E, L
CA	1	R
CAR	4	E, I, D, E
CARE	4	F, S, D, L
CAREF	2	U, R
CAREFU	1	L
CAREFUL	1	BLANK

Using the complete word segmentation method, the test word "CAREFUL" will be segmented into "CARE" and "FUL," since CARE appears as a word in the corpus. The peak and plateau method would give the same result.

In summary, the successor variety stemming process has three parts:

(1) First part determines the successor varieties for a word,

(2) The information provided in the first step is used to segment the word with any of the method described above, and

(3) Finally to select one of the segments as the stem.

2.2.3 Table Lookup Method

Terms and their corresponding stems can also be stored in a table. Stemming is then done via lookups in the table. One way to do stemming is to store a table of all index terms and their stems. Terms from queries and indexes could then be stemmed via table lookup [11]. Using B-tree or Hash table, such lookups would be very fast. For

example, presented, presentable, presenting all can be stemmed to a common stem present.

There are problems with this approach. The first is that there for making these lookup tables we need to extensively work on a language. There will be some probability that these tables may miss out some exceptional cases. Another problem is the storage overhead for such a table.

2.2.4 N- Gram Method

Another method of conflating terms called the shared digram method given in 1974 by Adamson and Boreham [15]. A digram is a pair of consecutive letters. Besides digrams one can also use trigrams and hence it is called n-gram method in general [16]. In this approach, pairs of words are associated on the basis of unique digrams they both possess. For calculating this association measures one use Dice's coefficient [11]. For example, the terms information and informative can be broken into digrams as follows.

information => in nf fo or rm ma at ti io on
unique digrams = in nf fo or rm ma at ti io on
informative => in nf fo or rm ma at ti iv ve
unique digrams = in nf fo or rm ma at ti iv ve

Thus, "information" has ten digrams, of which all are unique, and "informative" also has ten digrams, of which all are unique. The two words share eight unique digrams: in, nf, fo, or, rm, ma, at, and ti.

Once the unique digrams for the word pair have been identified and counted, a similarity measure based on them is computed. The similarity measure used is Dice's coefficient, which is defined as: $S = \frac{2C}{A + B}$

where *A* is the number of unique digrams in the first word, *B* the number of unique digrams in the second, and *C* the number of unique digrams shared by *A* and *B*. For the example above, Dice's coefficient would equal (2 x 8) / (10 + 10) = .80. Such similarity measures are determined for all pairs of terms in the database. Once such similarity is computed for all the word pairs they are clustered as groups. The value of

Dice coefficient gives us the hint that the stem for these pair of words lies in the first unique 8 digrams.

2.3 Classification of Stemming Algorithm

Stemming algorithms can be broadly classified into two categories, namely Rule – Based and Statistical.

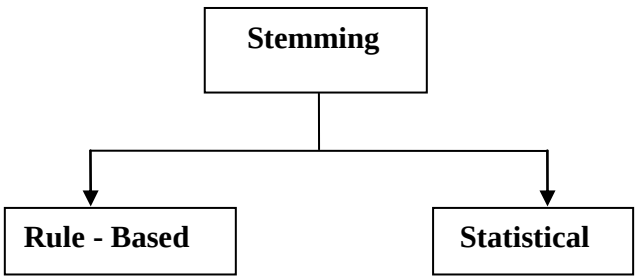

Figure 2.2 Types of Stemming Approach

Rule based stemmer encodes language specific rules where as statistical stemmer employs statistical information from a large corpus of a given language to learn the morphology.

2.3.1 Rule Based Approach

In a rule based approach language specific rules are encoded and based on these rules stemming is performed. In this approach various conditions are specified for converting a word to its derivational stem, a list of all valid stems are given and also there are some exceptional rules which are used to handle the exceptional cases. In Lovins stemmer, stemming comprises of two phases [12]: In the first phase, the stemming algorithm retrieves the stem from a word by removing its longest possible ending by matching these endings with the list of suffixes stored in the computer and in the second phase spelling exceptions are handled. For example the word "absorption" is derived from the stem "absorpt" and "absorbing" is derived from the stem "absorb". The problem of the spelling exceptions arises in the above case when one tries to match the two words "absorpt" and "absorb". Such exceptions are handled very carefully by introducing recording and partial matching techniques in the stemmer as post stemming procedures.

Recording [12] occurs immediately following the removal of an ending and makes such changes at the end of the resultant stem as are necessary to allow the ultimate matching of varying stems. These changes may involve turning one stem into another (e.g. the rule $\text{rpt} \rightarrow \text{rb}$ changes absorpt to absorb), or changing both stems involved by either recording their terminal consonants to some neutral element ($\text{absorb} \rightarrow \text{absor}\partial$, $\text{absorpt} \rightarrow \text{absor}\partial$), or removing some of these letters entirely, that is, changing them to nullity ($\text{absorb} \rightarrow \text{absor}$, $\text{absorpt} \rightarrow \text{absor}$).

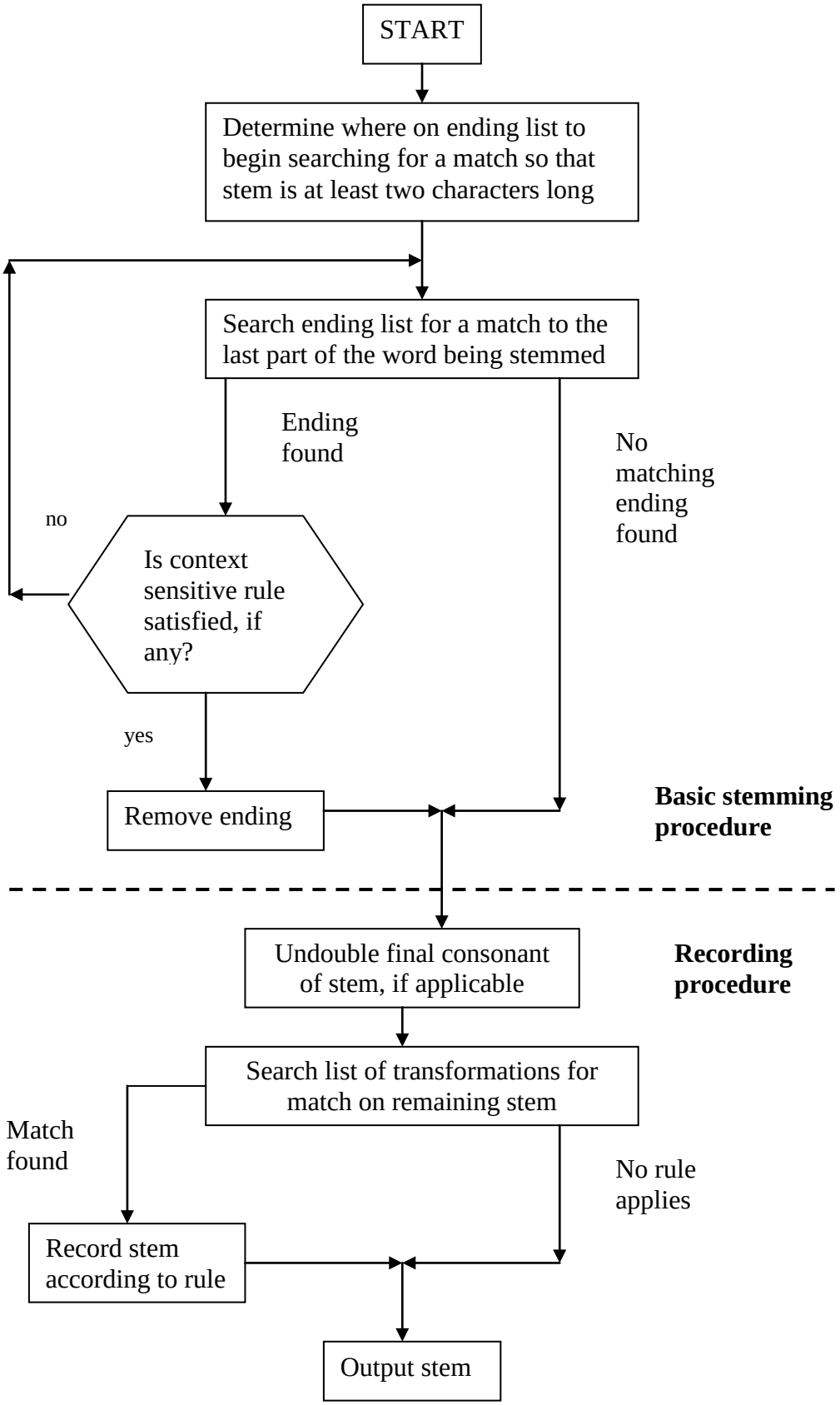

Figure 2.3 Flowchart Depicting Stemming and Recording Routines [12]

The main difference between recording and partial matching is that a recording procedure is a part of stemming algorithm whereas partial matching procedure is applied on the output of stemming algorithm where the stems derived from the catalogue terms are being searched for matches to the user's query.

Apart form Lovins method; one more rule based method is given by MF Porter which comprises of a set of conditional rules [13]. These conditions are either applied on the stem or on the suffix or on the stated rules. As per the conditions, a word can be represented in a general form like:

$[C]\,(VC)^{m}\,[V]$

Where C represents a list of consonants, V represents a list of vowels and m represents the measure of any word. For example:

m=0 RA, EE, BI, AT

m=1 TREES, OATS, RATES

m=2 TEACHER, TROUBLES, SITUATION

The general rule for removing a suffix is given as:

$(\text{condition})S1 \rightarrow S2$

Where, condition represents a stem and if the condition is satisfied then suffixes S1 is replaced by suffix S2. For example

$(m > 1)\text{ION} \rightarrow$

Here S1 is ION and S2 is null. This would map EDUCATION to EDUCAT, since EDUCAT is a word part for which m=2.

Advantages

1. Rule Based stemmers are fast in nature i.e. the computation time used to find a stem is lesser.

2. The retrieval results for English by using Rule Based Stemmer are very high.

Disadvantages

1. One of the main disadvantages of Rule Based Stemmer is that one need to have extensive language expertise to make them.
2. The procedure used in this approach handles individual words: it has no access to information about their grammatical and semantic relations with one another.
3. The amount of storage required to store rules for stem extraction from the words and also to store the exceptional cases.
4. These stemmers may apply over stemming and under stemming to the words.

2.3.2 Statistical Approach

Statistical stemming is an effective and popular approach in information retrieval [17] [18]. Some recent studies [19] [20] show that statistical stemmers are good alternatives to rule-based stemmers. Additionally, their advantage lies in the fact that it does not require language expertise. Rather it employs statistical information from a large corpus of a given language to learn morphology of words. A lot of research has been done in the area of statistical stemming method, some of the latest works are stated below:

2.3.3 Yet Another Suffix Stripper (YASS)

Most popular stemmers encode a large number of languages specific rules built over a length of time. Such stemmers with comprehensive rules are available only for a few languages. In the absence of extensive linguistic resources for certain languages, statistical language processing methods have been successfully used to improve the performance of IR systems. Yet another suffix stripper (YASS) is one such statistics based language independent stemmer [20]. Its performance is comparable to that of Porter's and Lovin's stemmers, both in terms of average precision and the total number of relevant documents retrieved the challenge of retrieval from languages with poor resources.

In this approach, a set of string distance measures [21] is defined, and complete linkage clustering is used to discover equivalence classes from the lexicon. The string distance measure is used to check the similarity between two words by calculating the

distance between two strings , the distance function maps a pair of string a and b to a real number r, where a smaller value of r indicates greater similarity between a and b. A set of string distance measures {D_1, D_2, D_3, and D_4} for clustering the words. The main reason to calculate these distances is to find long matching prefixes and to penalize an early mismatch.

Given two strings $X = x_0x_1.....x_n$ and $Y = y_0y_1.....y_{n'}$ we first define a Boolean function p_i as penalty for an early mismatch:

$$p_i = \begin{cases} 0 \text{ if } x_i = y_i \quad 0 \le i \le \min(n,n') \\ 1 \text{ otherwise} \end{cases}$$

Thus, p_i is 1 if there is a mismatch in the ith position of X and Y. If X and Y are of unequal length, a shorter string will be padded with null characters to make its length equal to the other string. Let the length of the string be n+1. We define D_1 as follows:

$$D_1(X, Y) = \sum_{i=0}^{n} \frac{1}{2^i} p_i \qquad (1)$$

Accordingly D_2, D_3 and D_4 are defined as follows:

$$D_2(X, Y) = \frac{1}{m} \times \sum_{i-m}^{n} \frac{1}{2^{i-m}} \text{ if } m > 0, \infty \text{ otherwise} \qquad (2)$$

$$D_3(X, Y) = \frac{n - m + 1}{m} \times \sum_{i-m}^{n} \frac{1}{2^{i-m}} \text{ if } m > 0, \infty \text{ otherwise} \qquad (3)$$

$$D_4(X, Y) = \frac{n - m + 1}{n + 1} \times \sum_{i-m}^{n} \frac{1}{2^{i-m}} \qquad (4)$$

Where, m represents the position of first mismatch between X and Y. Figure 2.4, considers two pair of strings {independence, independently} and {indecent, independence} and value of various distance measure for these two pair of words is calculated as below. Clearly one can infer that indecent and independent are farther apart from independence and independently.

0	1	2	3	4	5	6	7	8	9	10	11	12
I	N	D	E	P	E	N	D	E	N	C	E	*
I	N	D	E	P	E	N	D	E	N	T	L	Y

$D_1 = \frac{1}{2^{11}} + \frac{1}{2^{12}} = 0.00073$

$D_2 = \frac{1}{11} \times (\frac{1}{2^0} + \frac{1}{2^1}) = 0.1363$

$D_3 = \frac{2}{11} \times (\frac{1}{2^0} + \frac{1}{2^1}) = 0.2727$

$D_4 = \frac{2}{13} \times (\frac{1}{2^0} + \frac{1}{2^1}) = 0.2307$

Edit Distance = 2

0	1	2	3	4	5	6	7	8	9	10	11
I	N	D	E	C	E	N	T	*	*	*	*
I	N	D	E	P	E	N	D	E	N	C	E

$D_1 = \frac{1}{2^4} + \frac{1}{2^5} + ... + \frac{1}{2^{11}} = 0.1245$

$D_2 = \frac{1}{4} \times (\frac{1}{2^0} + \frac{1}{2^1} + ... + \frac{1}{2^{11-4}}) = 0.4980$

$D_3 = \frac{8}{4} \times (\frac{1}{2^0} + \frac{1}{2^1} + ... + \frac{1}{2^{13-11}}) = 3.984$

$D_4 = \frac{8}{12} \times (\frac{1}{2^0} + \frac{1}{2^1} + ... + \frac{1}{2^{13-11}}) = 1.328$

Edit Distance = 8

Figure 2.4 Calculations of Distance Measures

This distance counts the minimum number of edit operations (inserting, deleting, or substituting a letter) required to transform one string to the other. Once similarity between pair of words have been calculated using distance measure, cluster of the words are made by using complete linkage algorithm. In the complete-linkage algorithm [22], the similarity of two clusters is calculated as the minimum similarity

between any member of one cluster and any member of the other, the probability of an element merging with a cluster is determined by a least similar member of the cluster.

2.3.4 Graph Based Stemmer (GRAS)

GRAS is a graph based language independent stemming algorithm for information retrieval [23]. The following features make this algorithm attractive and useful: (1) retrieval effectiveness, (2) generality, that is, its language-independent nature, and (3) low computational cost. The steps that are followed in this approach can be summarized as below:

1. Find long common prefix among the word pairs present in the documents. For this, consider the word-pairs of the form $W_1 = PS_1$ & $W_2 = PS_2$ where, P is the long common prefix between W_1 & W_2.
2. The suffix pair S_1 & S_2 should be valid suffixes i.e. if other word pairs also have a common initial part followed by these suffixes such that $W'_1 = P'S_1$ & $W'_2 = P'S_2$. Then, S_1 & S_2 is the pair of candidate suffix if large number of word pairs is of this form. Thus, suffixes are considered in pair rather than individually.
3. Look for pairs that are morphological related i.e. if
   - They share a non-empty common prefix.
   - The suffix pair is a valid candidate suffix pair.
4. These words relationships will be modelled using a Graph where nodes represent the words and edges are used to connect the related words.
5. Pivot node is identified i.e. pivot is considered that node which is connected by edges to a large number of other nodes.
6. In the final step, a word that is connected to a pivot is put in the same class as the pivot if it shares many common neighbours with the pivot.

Once such words classes are formed, stemming is done by mapping all the words in a class to the pivot for that class. This stemming algorithm has outperformed Rule-Based Stemmer, Statistical Stemmer (YASS, Linguistica [24] etc), and Baseline Strategy.

Advantages

1. Statistical stemmers are useful for languages having scarce resources. Like the Asian languages are heavily used in Asian Sub Continent but very less research is done on these languages.
2. This approach yields best retrieval results for suffixing languages or the languages which are morphologically more complex like French, Portuguese, Hindi, Marathi, and Bengali rather than English.
3. They are considered as Recall – Enhancing Devices as they increase the value of recall at a given rate.

Disadvantages

1. Most of the statistical stemmer does their statistical analysis based on some sample of the actual corpus. As sample size decreases, the possibility of covering most morphological variants will also decrease. Naturally, this would result in a stemmer with poorer coverage.
2. For the Bengali lexicon, there are few instances where two semantically different terms fall in the same cluster due to their string similarity. For example, *Akram* (the name of a cricketer from Pakistan) and *akraman* (to attack) fall in the same cluster, as they share a significant prefix. Such cases might lead to unsatisfactory results.
3. Statistical Stemmers are time consuming because for these stemmers to work one needs to have complete language coverage, in terms of morphology of words, their variants etc.

2.4 Comparison Among Stemming Approaches

This section compares the performance of various stemming approaches discussed till now. This comparison considers one rule-based approach and compares it with statistical approaches like YASS and GRAS. The parameters used in this comparison are each stemmer's strength and the computation time required by each stemmer to compute the stem.

2.4.1 Stemmer Strength

Stemmer Strength [25] generally represents the extent to which a stemming method changes words to its stems. One well-known measure of stemmer strength is the average number of words per conflation class. Formally, if Na, Nw, and Ns denote the mean number of words per conflation class, the number of distinct words before stemming and the number of unique stems after stemming respectively, then $Na = \frac{N_w}{N_s}$ [19].

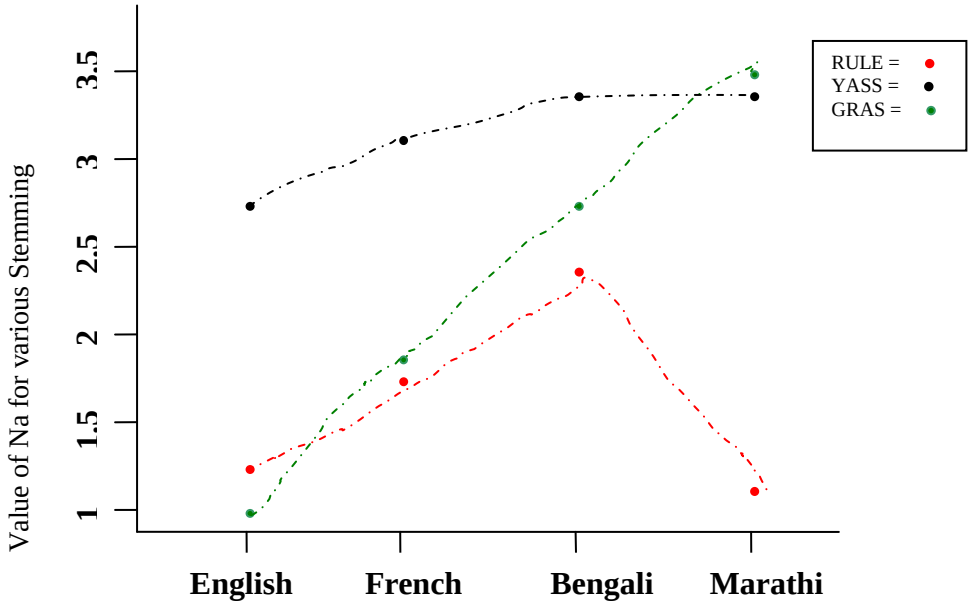

Figure 2.5 Stemmer Strength

Figure 4 gives the value of N_a for various stemming methods, clearly a higher value of N_a indicates a more aggressive stemmer. Among the three stemmers discussed above, YASS appears to be particularly aggressive on all languages and produces largest N_a value for English, French and Bengali. On the other hand, GRAS is the most aggressive on Marathi while it works equally well as rule- based stemmer for other languages like English, French and Bengali.

2.4.2 Computation Time

The comparison above clearly shows that YASS outperforms all other stemmer. One more parameter that is used by researchers for comparing the performance of stemmers is computation time which includes the time from submitting a query to its processing and final retrieval. Figure 5 clearly shows that for equal number of words in various languages like English, French, Bengali and Marathi the computation time of YASS is far more than its closest competitor GRAS [23]. This concludes that GRAS is far faster than YASS. In GRAS, two aspects that influence the processing time are the density of graph, that is, average degree of a node, and the length of the suffix.

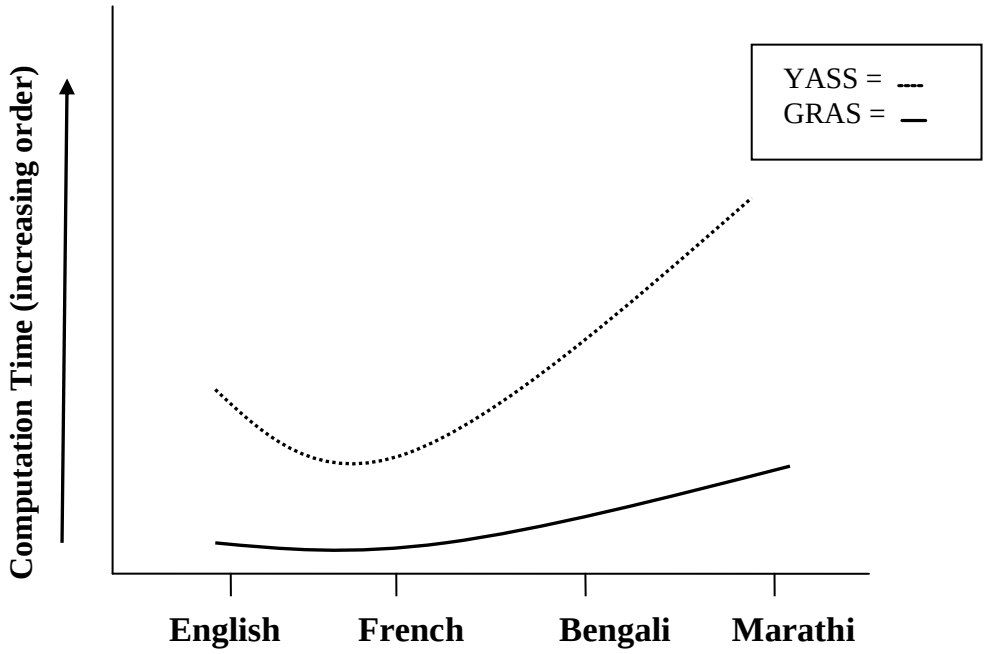

Figure 2.6 Computation Time

CHAPTER 3
PROBLEM STATEMENT

3.1 Problem Definition

As discussed in previous chapter, stemming greatly enhances the performance of information retrieval systems. There are number of approaches to perform stemming from rule based approach to statistical approach and graph based stemmer is one such statistical based stemmer which has surpassed the performance of previous stemmer to a greater extent but there are still some open issues that are to be dealt properly.

- The very first step of grouping the words present in the documents such that each group should have a common prefix of length at least a given threshold value ‘l’ seems theoretical in nature. In practical terms it is very difficult to scan through the complete document of hundred of pages and make such group of words having common prefixes.

- The approach used in the present stemmers to find the common prefixes is very time consuming where the stemmer has to scan all the words in the first pass and in the second pass it starts making group of those words. Thus it takes quadratic time to find the common prefixes among the words.

- Lexicon is considered to be already sorted which may not be the case practically.

3.2 Proposed Objective

The main objectives that are addressed in the thesis to solve the above mentioned problem are as follows:

- To study and compare the statistical stemming approach used for suffixing (inflectional) languages.

- To propose a new approach to find the common prefix among the words present in a document.

- To compare this new approach with those already being used to see how efficient it is as compared with other approaches.

- To verify and analyse the results in support of this proposal.

3.3 Methodology Used

- Study the various stemming approaches used to stem the words mainly for inflectional languages and make a comparison on them.

- Compare all the string matching algorithms based on their time and space requirement to see which alternative is best suited to our requirement of finding longest common prefix.

- Develop a new algorithm using the new approach

- Verify and analyse the behaviour of developed algorithm by considering various cases of different data sets.

CHAPTER 4
IMPLEMENTATION

4.1 Analysis of Existing Algorithm

The existing stemming algorithms are analyzed and compared on the basis of type, nature of stemming used, principle of working, various issues. And based on this following comparison table is made:

Table 4.1 Comparison Among Stemming Approaches

	Lovins Stemmer	**Porter Stemmer**	**YASS**	**GRAS**
Type of stemming	• Rule-Based/ Context Sensitive	• Rule-Based / Context sensitive suffix removal algorithm	• Statistical / Context free algorithm	• Statistical / Context free algorithm
Nature of Stemming	• Language Dependent	• Language Dependent	• Language Independent	• Language Independent
Principle of working	• Stored list of suffixes are used for matching the words. • Exceptions are handled using Recording and Partial matching	• Set of conditional rules are used to generate root words • It iteratively removes the suffixes from words until none of the rule applies	• String Distance measure is used to find related words and then complete linkage clustering is used to discover equivalence classes	• Common prefixes are used to generate valid suffix pairs and then set of classes of morphologically related words are constructed.
Issues	• suffers from over stemming of words	• Words having different meaning are reduced to same stem. • It ignores prefixes completely(reliable and unreliable remain unrelated)	•Finding equivalence is computationally expensive step.	• Finding common prefixes and valid suffix pair's is expensive step.

From the above comparison table one concludes that the stemming approach used in rule based method make their use very limited. The use of this approach is limited to those languages for which the stemming rules are already specified. Moreover, if some exception occurs to the rules then it has to be handled separately. Some of the rule based approaches may also show understemming or overstemming as a side effect of excessive use of rules.

On the other hand, statistical approach shows a wide usage. Statistical methods can be used to perform stemming for resource poor languages. In this approach, the words can be grouped under their common form by using clustering method or by making sets of different classes. The clustering method as used in YASS is computationally very expensive and complex. Another approach is to make classes of different words by finding suffix-pair among them. The second approach has been used in GRAS.

However, GRAS divides the stemming into two steps:

- The first step finds the common prefix among the different words and then extracts valid suffix-pairs from them.
- In the second step the valid suffix-pair frequency is used to make sets of different classes. For each class a pivot word (root word) is derived and this word is used as a stem.

This thesis has suggested an improvement in the first step of stemming i.e. to find valid suffix-pair. It has proposed an algorithm which is easy to implement and also finds the suffix pair is lesser time complexity.

4.2 Design of New Algorithm

The design of the new algorithm starts with constructing Trie for all the unique words. So first of all there is an introduction to Tries with the help of some examples. Then the basic node structure of Trie is explained with its various implementation alternatives and finally the improved algorithm is proposed. In the following discussion the terms strings and words are used interchangeably.

4.2.1 Trie

Trie [27] is an ordered tree data structure that is used to store strings over an alphabet. Unlike a binary search tree, no node in the tree stores the key associated with that node; instead, its position in the tree shows what key it is associated with. Each node contains an array of pointers, one pointer for each character in the alphabet and all descendants of a node have a common prefix of the string associated with that node. The root is associated with the empty string and values are normally not associated with every node, only with leaves.

The main property of trie that is being used in the proposed algorithm is that tries allows words with similar character prefixes to use the same prefix data and store the rest of the word in the form of tails or child nodes as a separate data [32]. One character of the string is stored at each level of the tree, with first character of the string stored at the root. The term trie comes from retrieval.

For example, in the case of English alphabetical keys, each node will represent an array of (27) pointers to its branches, where the first 26 pointers are used for each alphabet character and the last one for blank (“”). The actual keys are stored in leaf (information) nodes.

Figure 4.1 illustrates an example trie for alphabetical keys. The trie stores the keys ASCENDING, BISCUIT, BIRTH, BUS, CARE, CAREFUL, CAREER, and TROY.

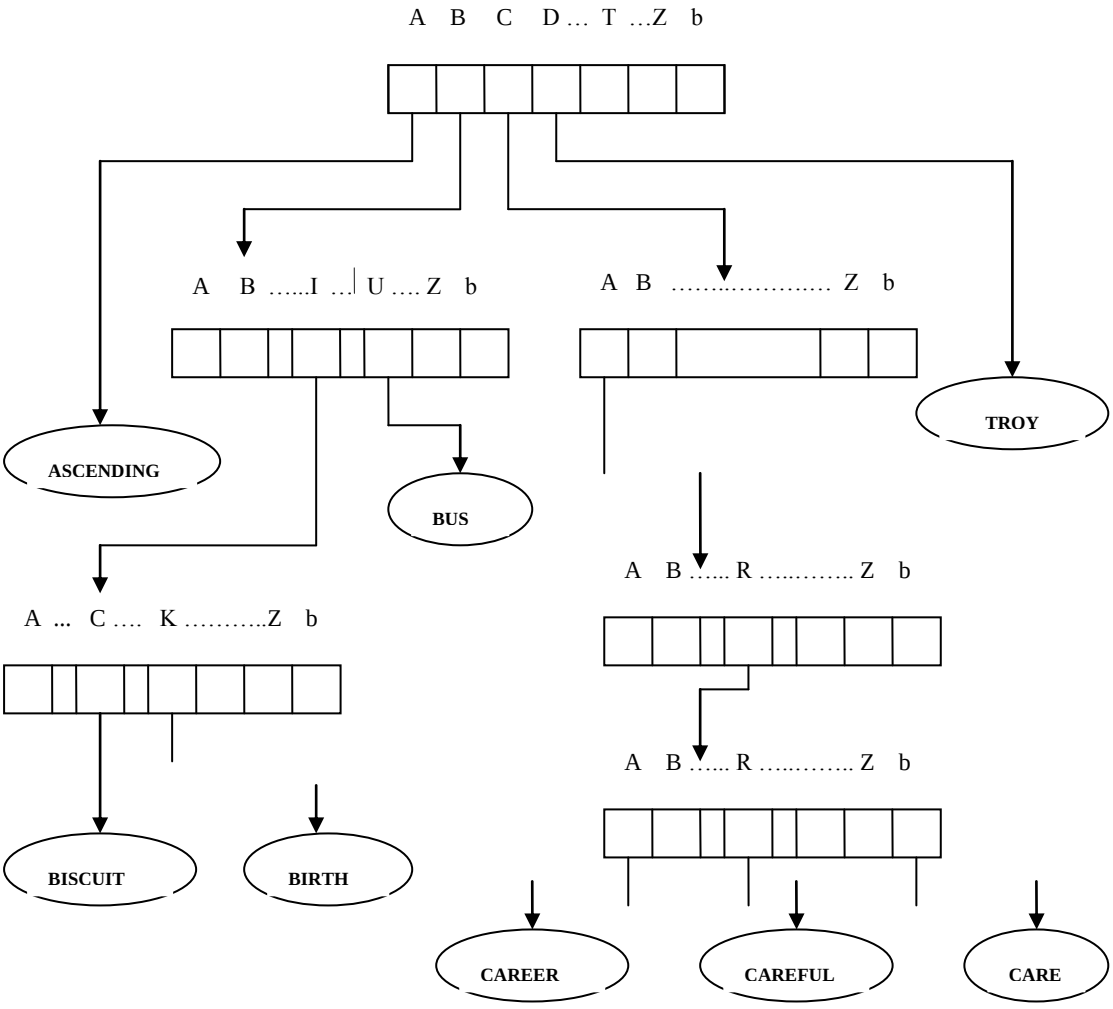

Figure 4.1 An Example of Trie

To access these information nodes, one follows a path beginning from a branch node moving down each level depending on the character forming the key, until the appropriate information node holding the key is reached. Thus the depth of an information node in a trie depends on the similarity of its first few character (prefix) with its fellow keys. Here, while ASCENDING and TROY occupy shallow levels (level 1 branch node) in the trie, CARE, CAREER, CAREFUL have moved down by level 4 levels of branch node due to their uniform prefix "CAR". The role played by the blank field in the branch node is evident when we move down to access CAR. While the information node pertaining to CAR positions itself under the blank field,

those of CAREER and CAREFUL having CARE as their common prefix will attach themselves to pointers from E to F respectively of the same branch node.

Each node of the trie needs to store a key (usually a string – here an array of chars) and an array of pointers to its branches. The branches correspond to the 26 results that can be returned by the character position in alphabet with respect to ‘a’ and the blank character. And each node has a variable called as “NotLeaf” to mark the end of a word- indicates if the node is information or an intern node.

The basic element- Node of a trie data structure looks like this:

```
Typedef struct trie_node {
        bool NotLeaf;
        trie_node *pChildren [NR];
        var_type word [20];
} node;
Where:
#define NR 27                    // the American alphabet (26 letters) plus blank
typedef char var_type;           //the key is a set of characters
```

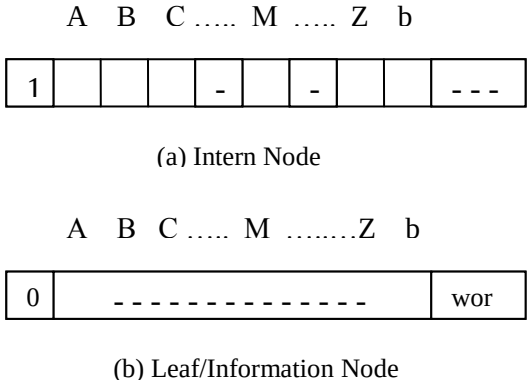

Figure 4.2 Internal and Leaf Node Structure of a Trie [32]

The construction for a trie_node simply sets all pointers in the node to NULL; for intern nodes NULL key is used and for leaves the desired string is stored in word[].

trie_node *NewIntern();
trie_node *Leaf(char word[]);

The main abstract methods for the TRIE ADT are;

1. bool search (char string[]);
2. void insert (char string[]);

Advantages

1. The common prefix among various words can be found in O(m), where m is the length of the common prefix.
2. Use of Trie helps in the grouping of various words sharing common prefix in less time.

4.2.2 The Proposed Approach

This section discusses that how the idea of using Trie as a data structure for storing the words will help in finding the common prefixes among them. Further it will show that the time required to look for the common prefix among various words is O (m), where m is the length of the common prefix. For this we need to consider the following lexicon of words and construct Trie for them:

care, prepare, careful, cared, preparing, preparation, caring

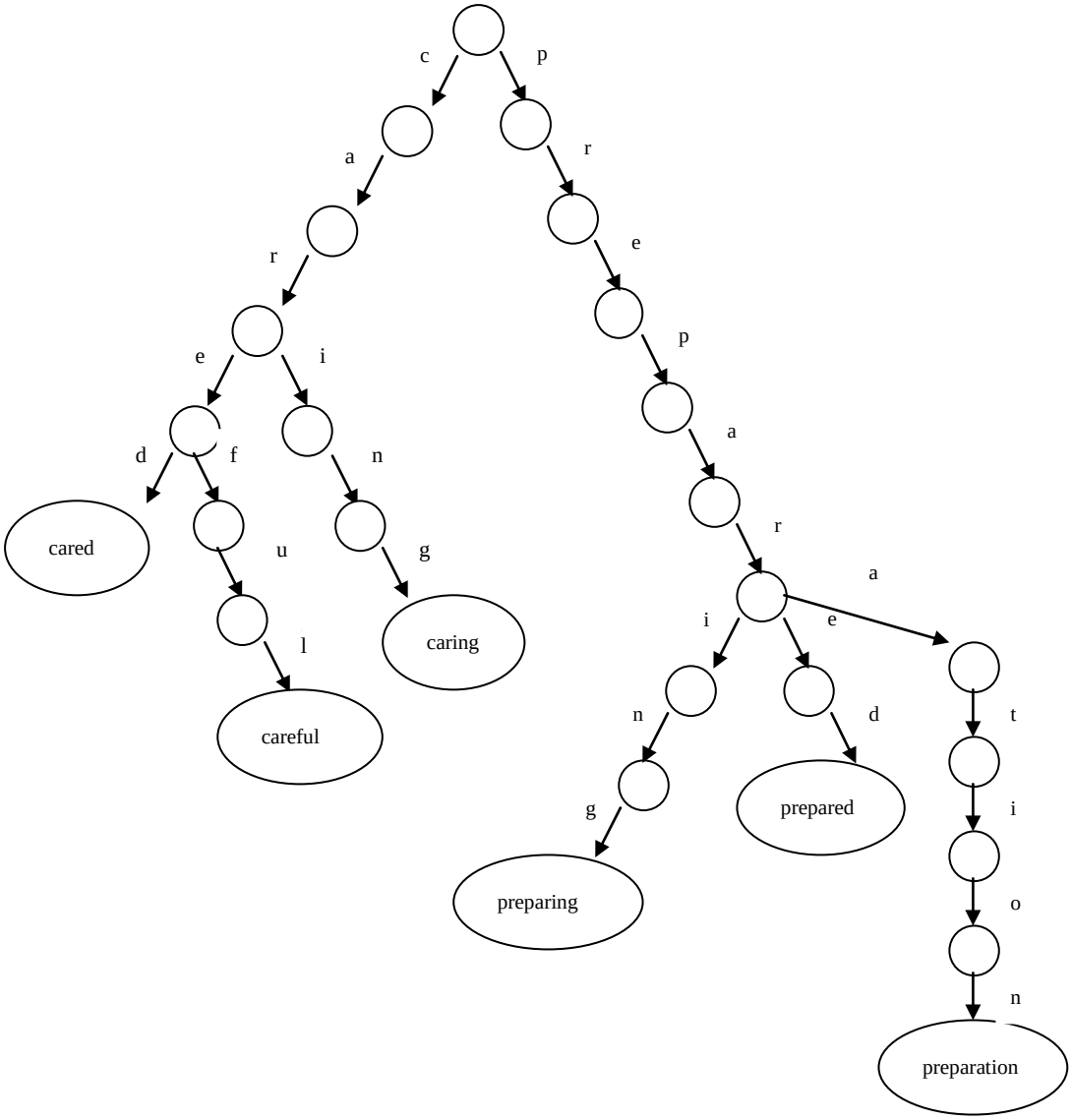

Figure 4.3 Elaborate Trie Structure for Words: care, prepare, careful, cared, preparing, preparation, caring

Now to find the common prefix among these words we need to traverse the Trie in the Depth First Search order. The words with the common prefix are branched out from the same parent node. Such parent nodes will give the common prefixes that have the potential of being the valid common prefix. While traversing down the Trie one needs to search for those nodes that have more than one child node. Once such nodes are found the parent of such nodes are considered to be potential common prefix. The same idea can be shown from the following figures:

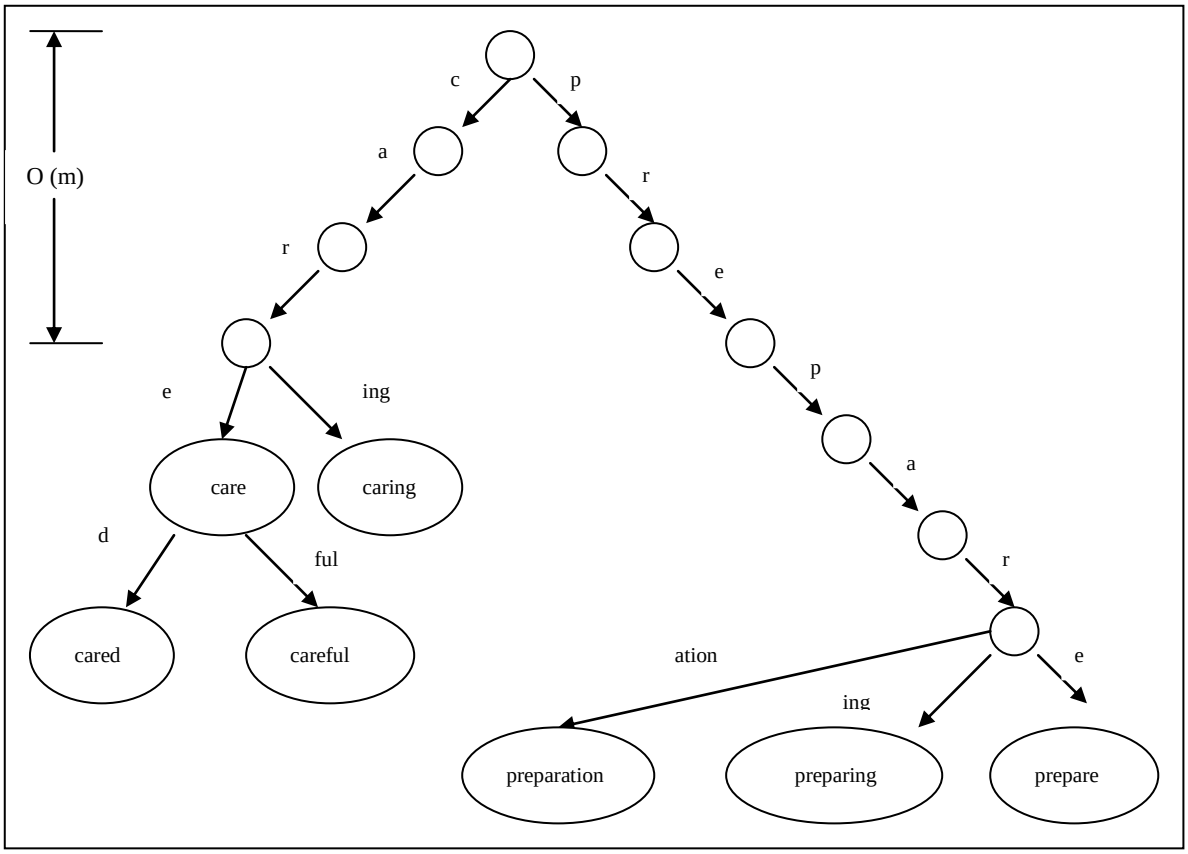

Figure 4.4 Branched Out Words from Their Common Prefix

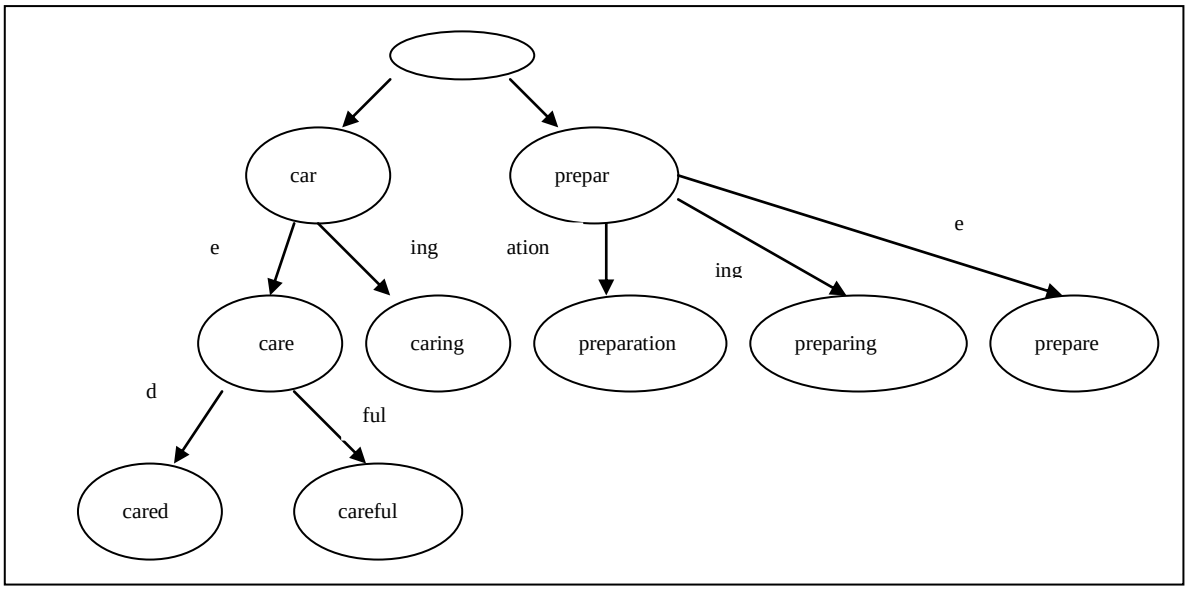

Figure 4.5 Compressed Trie Structure

4.2.3 Complexity Analysis:

Time taken to search for a common prefix or for a string in Trie and the space requirement for such structure depends on the implementation of the internal/child nodes.

Table 4.2 Comparison of Complexity Analysis of Various Trie-Node Implementations [28]

Implementation Choice	Time Required	Space Required
Array Per Node	O (m)	O (n*k)
Tree Per Node	O (m log k)	O (n)
Hash Children	O (m)	O (n)
Linked List	O (m*k)	O (n*k)

Where,

n = no of nodes in the Trie

k = total number of alphabets used in the Trie

m =number of alphabets in the string to be searched

For English alphabets value of k remain constant i.e. 26 so the space requirement for the Tries remain O (n*26) irrespective of its implementation. The time requirement for array, hash children, linked list implementation is comparable but array is preferred over others because of its easy implementation. There will be no case of collisions as in hash children and no complex handling of pointers as in linked list implementation.

4.2.4 The Improved Algorithm

The approach of stemming will be divided into two algorithms, firstly to group the words based on their common prefixes so that one can generate the valid suffixes from them. The second algorithm uses the result of first algorithm to generate the classes of morphologically related words.

There are two threshold values one for the common prefix length (l) and the other for the frequency of valid suffix-pair (α) that are compared in this algorithm. J H Paik [23] has suggested that value of both l and α should lie from 3 to 5.

4.2.5 Algorithm to Find Valid Suffixes

1. Remove the stop words and extra spaces from the documents so as to get a unique list of words.
2. Construct Trie for all the unique words in the document and assign each node with a value num_child which gives the number of children each node has.
3. Traverse down the Trie structure in DFS order and look for nodes the have more than one children.
4. If the path from the root to the nodes with num_child>1 is greater then the threshold value 'l' then it will give us the common prefix.
5. Arrange these nodes in decreasing order of their value of num_child. Make a list of these nodes called as list of pivot nodes p.
6. Construct a weighted graph for trie constructed above:
   - Consider the list of pivot nodes p = {p1, p2…pn}.
   - for (i=1 to n)

     - Create edges from pi to all its child nodes, where the edges represent candidate suffix-pair.
     - The suffix pair S_1 & S_2 should be valid suffixes i.e. if other word pairs also have a common initial part followed by these suffixes such that $W'_1 = P'S_1$ & $W'_2 = P'S_2$. Then, S_1 & S_2 is the pair of valid suffix if their frequency exceeds the threshold value α . Thus, suffixes are considered in pair rather than individually.
     - Compute the frequency of all suffix-pair
7. Find the set of classes by decomposing the graph using [23].

4.2.6 Flowchart

The above discussed algorithm can be depicted with the help of the following algorithm where the first flowchart explains how trie can be used to find common prefixes and second flowchart explains how to the valid suffixes are generated while construction a graph from trie.

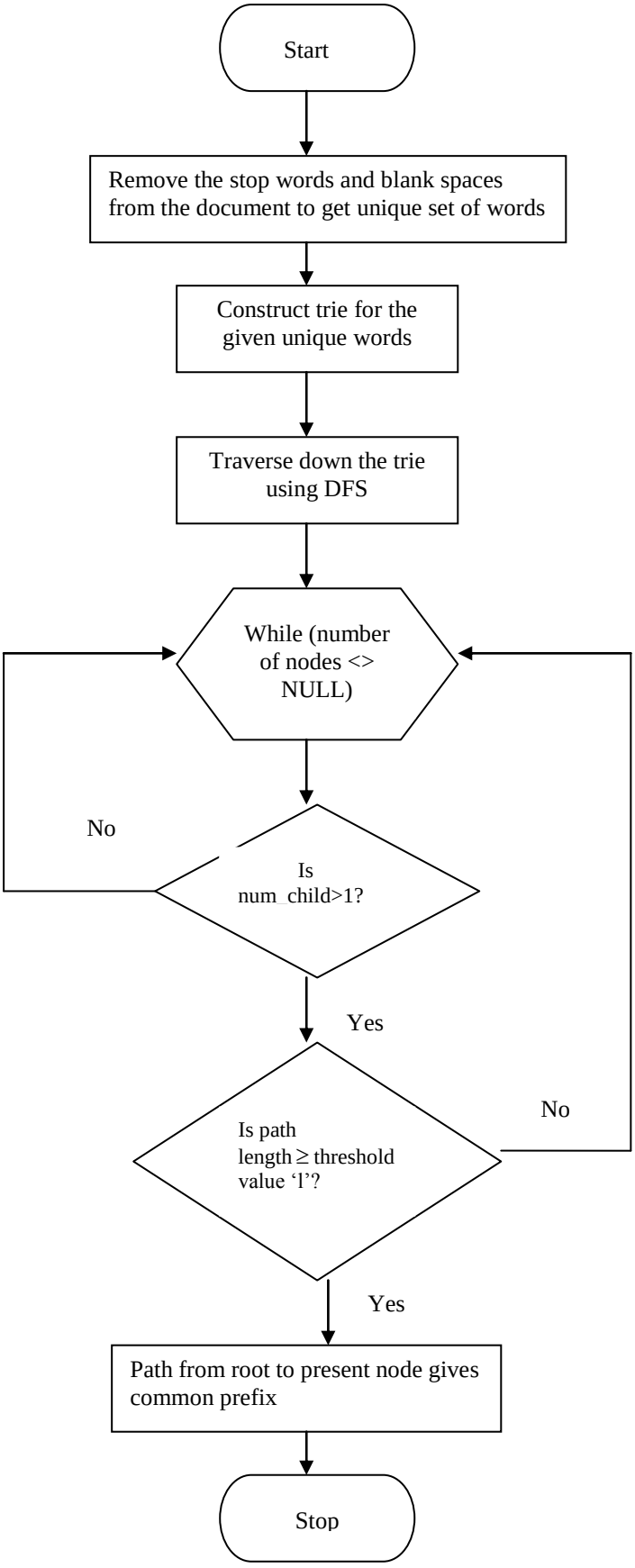

Figure 4.6 Flowchart to Find Common Prefixes Using Trie

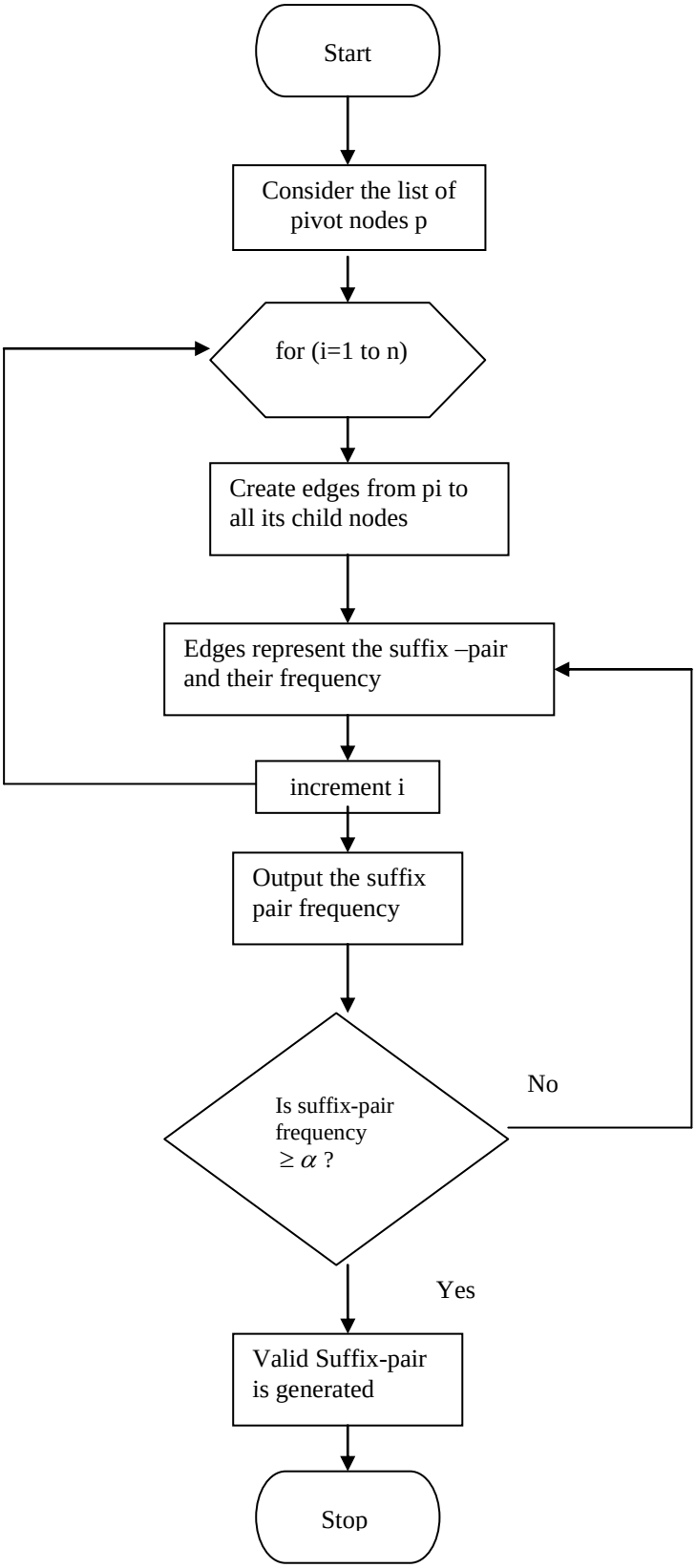

Figure 4.7 Flowchart to Generate Valid Suffix-Pair

4.3 Proof of Correctness for Finding the Common Prefix Among the Given Words in O (m).

Input: A Trie structure of unique words from a document

Output: Finding common prefixes among the words in O (m)

Proof by Mathematical Induction

Basis Step: In the base step only a pair of words is considered and it is seen that their common prefix can be found in O (m). Consider the pair (care, careful) and draw trie for them.

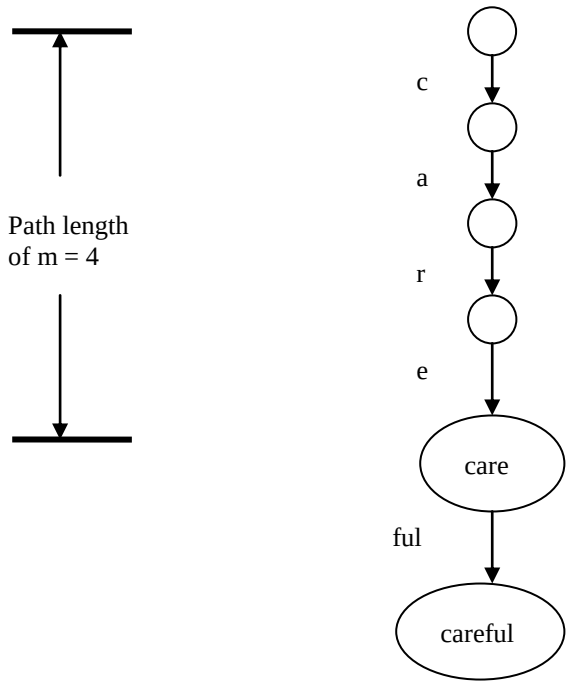

Figure 4.8 Basis Step for a Base Pair (care, careful)

Induction Hypothesis: In induction hypothesis consider that the problem of finding common prefix among words can be solved in O (m).

Inductive Step: The above hypothesis is proved for n+1 of words. Consider the following pairs of data sets: and, add, added, leg, legal, legally, setting, set, legged, addition, legging

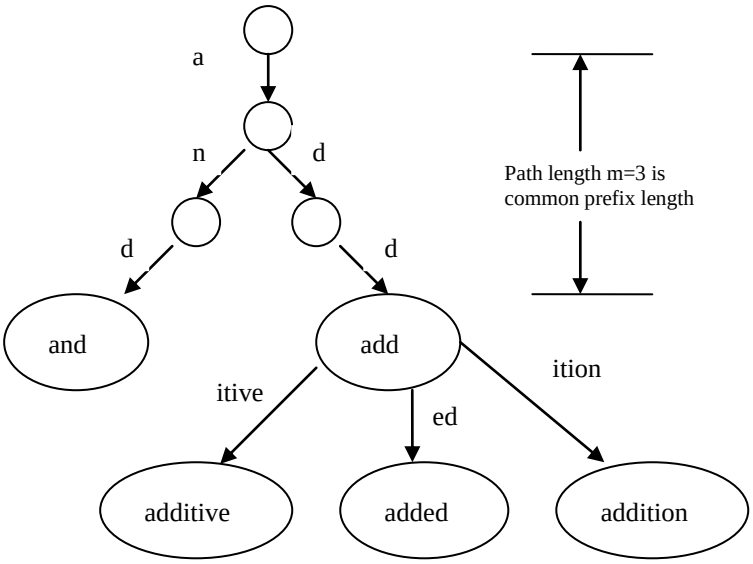

Figure (a)

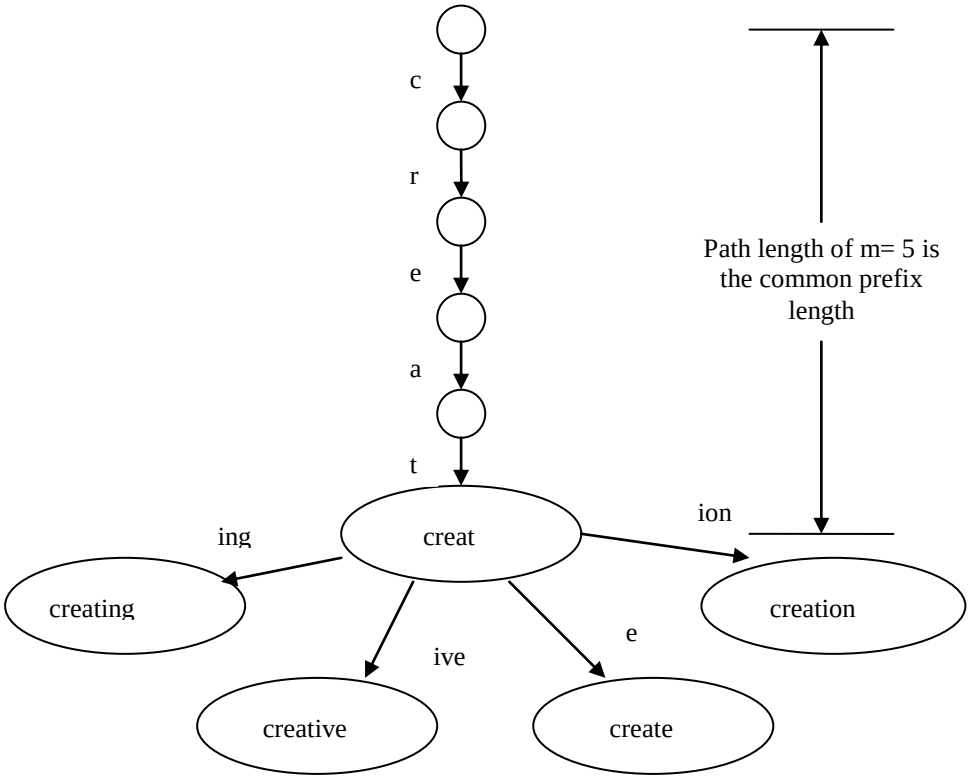

Figure (b)

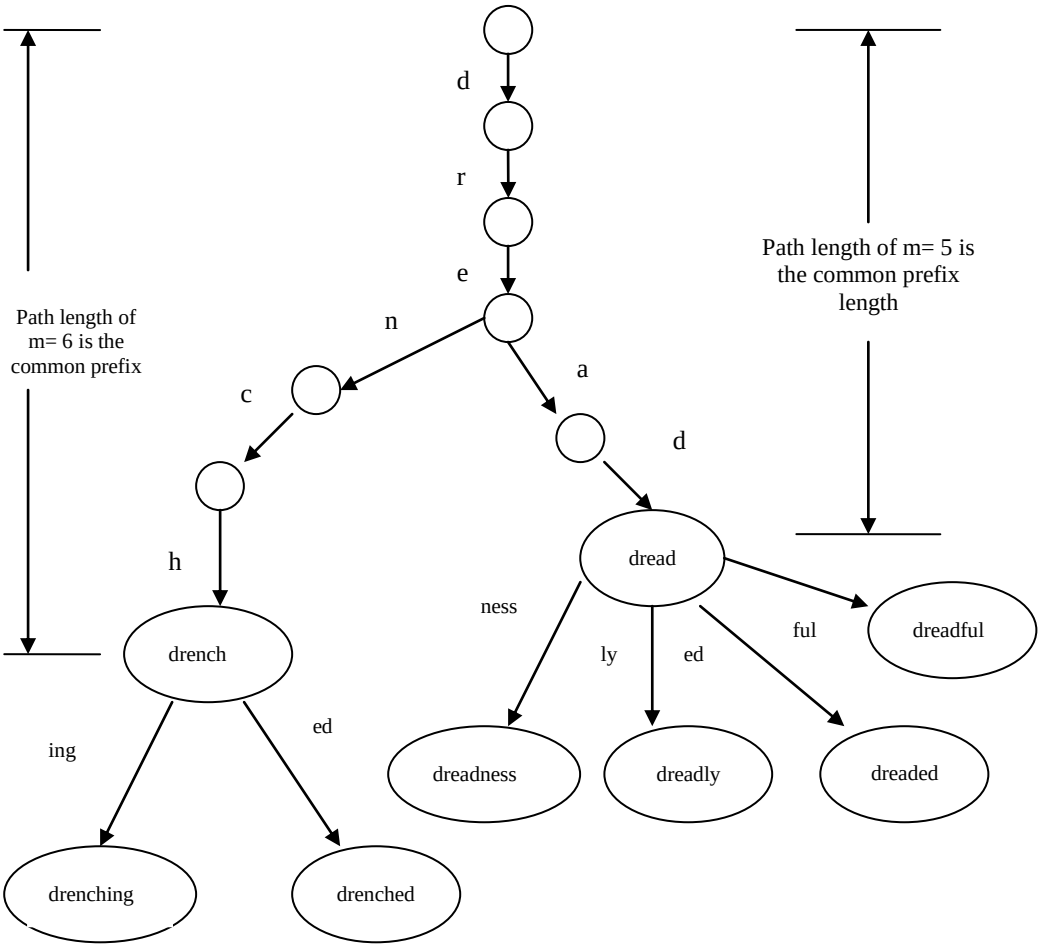

Figure (c)

Figure 4.9 (a, b, c) Inductive Step

From the various cases considered in inductive step one concludes that the complexity of finding common prefix among various words can be done in O (m) where m is the length of common prefix among the given words. Thus the inductive hypothesis holds true.

4.4 Analysis of the Proposed Algorithm

Before starting the complexity analysis of this algorithm consider following notations:

‘n’ = the number of words in a document.

'$\sum$' = the total number of alphabets used to construct words since we are considering English language words then '$\sum$' = k = 26

'm' = length of the common prefix

Table 4.3 Step-wise Analysis of Improved Algorithm

# Step	Analysis of Each Step	Time Requirement
Step 1	To remove Stop Words and Blank Spaces, single linear pass over the lexicon is required	**O (n)**
Step 2	Construct Trie for the unique words, requires linear time	**O(n)**
Step 3	Trie traversal using DFS, worst case complexity of DFS for traversal of complete tree is O (|V| + |E|) where V = no. of vertices and E= no. of edges. Here V= $\sum$ * n and E= $\sum$ * n -1	**O(|nk|+|(n-1)k|)**
Step 4	Computing the common prefix by traversing from root node to required node, these prefixes are valid if their length >= 'l', l is predefined threshold value.	**O(m)***
Step 5	Arranging nodes in decreasing order of their value	**O(n log n)****[29]
Step 6	Graph construction from trie, in linear time	**O(n)**
Step 7	Creating set of classes by decomposing the graph	**O(n)** [23]

*using proof of correctness

****using tree sorting algorithm [29]

4.5 Total Complexity

The total time complexity can be obtained by performing summation of time complexity of each step. The individual time complexity of each step is given in table 4.3 and their summation gives us T(n)

$$T(n) = c_1 n + c_2 n + c_3 (|nk + (n-1)k|) + c_4 m + c_5 (n\log n) + c_6 n + c_7 n$$

Where c_1, c_2, c_3, c_4, c_5, c_6, c_7 represents the cost incurred in each step

$T(n) = c_1 n + c_2 n + c_3 n k + c_3 n k - c_3 k + c_4 m + c_5 n\log n + c_6 n + c_7 n$

$T(n) = (c_1 + c_2 + c_6 + c_7) n + 2 * c_3 nk - c_3 k + c_4 m + c_5 n \log n$

Here c_1 , c_2 , c_6 , c_7 being the constant cost terms are merged to c_i . Also $2 * c_3$ are substituted by another constant c_j . Similarly c_3 , c_4 and c_5 are replaced by another constant terms like c_k , c_l , c_m .

$T(n) = c_i n + c_j nk - c_k k + c_l m + c_m n \log n$

$T(n) = c_n (n + nk) - c_k k + c_l m + c_m n\log n$

c_i and c_j being constant terms are replaced by single constant c_n . Here k represents the total number of character used to make different words, for English language value of k = 26 which is again constant. Also for any language the total number of such alphabets remains constant. So, the term $c_k k$ is a constant term and even if it is subtracted from the other factors the resulting changes can be ignored.

In the first term $c_n (n + nk)$ can be rewritten as $c_n (n (1 + k))$ and further as $c_n n$ because 1+ k is again a constant.
Thus the total cost in terms of time will be given as

$T(n) = c_n n + c_l m + c_m n \log n$

Out of all the three terms in above equation $n \log n$ is the largest. So, the time complexity in terms of Big-O notation is

$T(n) = O(n \log n)$

This time complexity is far more superior to the previously proved algorithms [23] where it is $O(n^2)$.

CHAPTER 5
CONCLUSION AND FUTURE SCOPE

Internet is a vast source of information and to harness this information various information retrieval systems are used. These systems can range from a classical information system like library system to a web based system like search engines. The effectiveness of the retrieval results can be judged by the number of relevant documents retrieved for any particular user request. To increases the effectiveness of results the documents needs to be pre-processed and undergo text processing operations. One such text processing operation which converts all the inflectional words to their base form is stemming. It has been studied that stemming increases the recall results of the retrieval performance [30] of any information retrieval system. It is also used to maintain index table by reducing the index table entries.

Section 2.3 discussed some of the stemming approaches used till date. These stemming approaches range from being rule based to statistics based. Rule based approach is mainly used for some specific language where as statistical based approach is language independent. Statistical Approach is mainly used for resource poor languages. Each such approach has its own advantage like Porter's stemmer shows high recall results for English language [13]. Its performance is comparable to the latest statistical based stemmers as discussed in section 2.3.3. But its main drawback is its language dependence. Similarly statistical stemmers like YASS, GRAS shows good results for inflectional / suffixing languages but again their performance for English is not that good. Also they are computationally expensive methods.

A comparison among all these approaches is made in section 4.1 on the basis of their nature, principle of working, issues. There is a proposal for new algorithm in section 4.2 which performs the step of finding suffix pair for stemming in O (n logn). It is an

improvement over previously used methods where this complexity is O (n^2) ('n' = the number of unique words present in the documents).

Future Scope

The approach to find valid suffix-pair among the given lexicon of words in O(n logn) is done by using tries as the base data structure but the same might be achieved by some other data structure and that too with better time complexity. However with the present approach one can further work on the following future aspects:

- The proposed algorithm can be used in text processing for other systems also like maintaining dictionary words, searching words in dictionary, maintaining library catalogue.
- The future scope in this case can be to improve the implementation of the proposed algorithm. One can also use other data structure like compressed trie to improve the results.
- Although string matching is not the direct application of the present approach but the algorithm can be modified to perform string matching also.

REFERENCES

[1] James M. Abello, Panos M. Pardalos, Mauricio G. C. Resende "Algorithmic Aspects of Information Retrieval on the Web" in "Handbook of Massive Data Sets", Kluwer Academic Publisher, 2002, pp 3-10.

[2] Djoerd Hiemstra "Information Retrieval Models" University of Twente Web: http:// www.cs.utwente.nl/~hiemstra

[3] Nicholas J. Belkin and W B Croft "Information filtering and information retrieval: two sides of the same coin" published in ACM- special issue on Information Filtering , Volume 35 Issue 12, Dec 1992

[4] Djoerd Hiemstra "Information Retrieval Modelling" Chapter 2 Website: http://comminfo.rutgers.edu/~aspoerri/InfoCrystal/Ch_2.html#2.3 accessed date 14th March, 2012

[5] J.Cho, H.Garcia-Molina, L.Page (1998). Efficient Crawling through URL Ordering. In (WWW7, 1998) pp 161-172

[6] S. Lawrence and C.L. Giles (1998) Searching the World Wide Web. Science, 280(5360):98.

[7] Ricardo Baesa Yates, Berthier Riberio-Neto "Modern Information Retrieval" chapter1

[8] A. Ramanathan and D. Rao, 2003." A lightweight stemmer for Hindi". In Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL), on Computational Linguistics for South Asian Languages (Budapest, Apr.) Workshop.

[9] J. Savoy 2008." Searching strategies for the Hungarian language". Inf. Process. Manage. 44, 1, 310–324.

[10] Robert Krovetz "Viewing morphology as an inference process" published in proceeding SIGIR'93 Proceedings of the 16th annual international ACM SIGIR conference on Research and development in IR

[11] WB Frakes, 1992, “Stemming Algorithm “, in “Information Retrieval Data Structures and Algorithm”, Chapter 8, page 132-139.

[12] J. B. Lovins 1968. "Development of a Stemming Algorithm." Mechanical Translation and Computational Linguistics, 11(1-2), 22-31.

[13] M. F. Porter 1980. "An Algorithm for Suffix Stripping Program", 14(3), 130-37.

[14] M. Hafer and S. Weiss 1974. "Word Segmentation by Letter Successor Varieties," Information Storage and Retrieval, 10, 371-85.

[15] G. Adamson and J. Boreham 1974. "The Use of an Association Measure Based on Character Structure to Identify Semantically Related Pairs of Words and Document Titles," Information Storage and Retrieval, 10, 253-60.

[16] P. McNamee, and J. Mayfield 2004.” Character n-gram tokenization for European language text retrieval”, Inf. Retr. 7(1-2), 73–97.

[17] J. Xu and W. B. Croft 1998.” Corpus-based stemming using co occurrence of word variants”. ACM Trans. Inf. Syst. 16, 1, 61–81.

[18] D.W. Oard, G.A. Levow and C.I. Cabezas 2001. CLEF experiments at Maryland:” Statistical stemming and back off translation”. In Revised Papers from the Workshop of Cross-Language Evaluation Forum on Cross-Language Information Retrieval and Evaluation (CLEF), Springer, London, 176–187.

[19] M. Bacchin, N. Ferro, and M. Melucci 2005. “A probabilistic model for stemmer generation”. Inf. Process. Manage. 41, 1, 121–137.

[20] P. Majumder, M Mitra, S.K. Parui, and G. Kole (ISI), P. Mitra (IIT), and K.K. Dutta. ”YASS: Yet another Suffix Stripper”, published in ACM Transaction on Information System (TOIS), Volume 25 Issue 4, October 2007, Chapter 18, Page 5-6.

[21] V. I. Levenstein 1966. Binary codes capable of correcting deletions, insertions and reversals. Commun. ACM 27, 4, 358–368.

[22] A. K. Jain, M.N. Murthy, and P. J. Flynn 1999. “Data clustering”: A review. ACM Comput. Surv. 31, 3, 264–323.

[23] JH Paik, Mandar Mitra, Swapan K. Parui, Kalervo Jarvelin, “GRAS : An effective and efficient stemming algorithm for information retrieval”, published in ACM Transaction on Information System (TOIS), Volume 29 Issue 4, December 2011, Chapter 19, page 20-24.

[24] J. Goldsmith 2001.” Linguistica: Unsupervised learning of the morphology of a natural language”. Comput. Linguist. 27, 2, 153–198.

[25] WB Frakes and C. J. Fox 2003. Strength and similarity of affix removal stemming algorithms. SIGIR.

[26] WB Croft, Donald Metzler, Trevor Strohman "Retrieval Models" Chapter 7 in book"Search Engines Information Retrieval in Practice", University of Massachusetts , Yahoo! Research, Google, 2009

[27] L. Allison "Tries" faculty of Information technology (Clayton), Monash University, AustraliaWebsite: http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Tree/Trie/ Accessed date 2nd April 2012

[28] R. Dorrigiv, D. Roche, C. Marriott, "Tries and String Matching", Data Structures and Data Management, School of Computer Science, University of Waterloo, 2010 table comparison

[29] M F Colton, P ferragina, S. Muthukrishnan "On the Sorting Complexity of Suffix Tree Construction" Published in Journal of ACM, Volume 47 Issue 6, Nov 2000

[30] M. Popovic and P. Willett. "The effectiveness of stemming for natural-language access to solven textual data". Journal of the American Society for Information Science, 43(5):383–390, 1992.

[31] WB Frakes 1984. "Term Conflation for Information Retrieval" in Research and Development in Information Retrieval, ed. C. van Rijsbergen. New York: Cambridge University Press.

[32] Website: http://software.ucv.ro/~cmihaescu/ro/laboratoare/SDA/docs/trie.pdf Accessed on 3rd April, 2012

LIST OF PUBLICATION

[1] Deepika Sharma & Deepak Garg "Information Retrieval on the Web and its Evaluation". International Journal of Computer Application (0975-8887), Vol. 40-No. 3, February 2012 (Published)