Designing and Developing a Lemmatizer for English Morphed Words handling Nominalization

A Synopsis submitted to

The Maharaja Sayajirao University of Baroda

in

Computer Science and Engineering

by

Mrs. Rupam Gupta

Research Guide

Dr. Anjali Ganesh Jivani

Department of Computer Science and Engineering
Faculty of Technology and Engineering

The Maharaja Sayajirao University of Baroda

Vadodara 390 001

March 2021

ABSTRACT

Morphology is the study of the principles and processes by which words are constructed in any language. So, morphological analysis is considered as a challenging task in the field of Natural Language Processing (NLP) and Text Mining (TM). Lemmatization is a major morphological analyzing operation that obtains the dictionary root word or lemma of an input morphed surface-word. It is a morphological transformation that changes a word as it appears in a running text into the base or dictionary form of the word by removing the inflectional and derivational ending of a word. Alternatively, stemming is a primitive way to obtain stem-token by chopping off the ends of any morphed word which are nothing but inflectional or derivational suffixes. After execution of any existing popular stemming process, it is observed that stems of morphed words are not necessarily dictionary words; they could be just string-token, representative of a group of allied morphed words. So these stems cannot be used as an informative entity in NLP or TM, since they do not make sense as compared to lemmas generated in lemmatization.

Stemming and lemmatization are the two most important pre-requisite tasks of most of the NLP, TM, Information Extraction (IE), Information Retrieval (IR) etc. applications. A better performing Lemmatizer is essentially required to find the correct dictionary root word for its any allied morphed word present in a text for text simplification, key-term identification, question answering systems, text summarization, topic identification, etc. for morphologically rich languages, where one root word might have many morphological variants due to agglutination or inflection. In an IE or IR system, lemmatization can provide support to improve overall retrieval recall since a query will be able to retrieve more relevant documents when variants in both query and documents are morphologically normalized.

This research work proposes a lemmatization model which is designed based on both, stemming and lemmatization techniques for obtaining the correct lemma from allied morphed words present in any input text. This model has significantly minimized the limitations of the currently available popular stemmers like Porter, Lovins, Paice, YASS etc. and lemmatizers like the Stanford-Lemma Processor, Spacy Lemmatizer, LemmaGen, WordNet Lemmatizer, Morph-Adorner etc. The existing lemmatizers generate lemma for all inflected English morphed words, like all singular-plural nouns, verb in different tenses where part-of-speech (POS) of morphed word and POS of root word are same, but all these lemmatizers

are not able to find out lemma for any type of derived words; specially for nominalized derived words where POS of derived word and root word are different. Nominalization or nominalisation is the use of a word form which actually does not exist as a noun but is being used as a noun or as the head of a noun phrase, with or without morphological transformation.

The proposed methodology extracts lemma for all allied morphed words like nouns, verbs, derivative words and especially nominalized words. The research work is a combination of language-independent statistical distance measures, segmentation technique, rule-based stemming approach and lastly morphological transducer's parsing technique to generate the correct dictionary word for a set of related morphed words. Through a single pass, the proposed model completes lemma generation process which leads to improve execution rate of the lemmatization process and correctly handles morphological nominalized words.

Two models have been designed, developed and implemented namely LemmaChase and LemmaQuest. LemmaChase being a lemmatizer which generates the correct lemma for individual morphed input words, whereas LemmaQuest being its superset, generates lemmas after creating clusters of related allied words. The work done has been compared with existing popular lemmatizers showing significant advanced lemma generation especially for nominalized words using the standard datasets available in the DUC-text Repository and the Oxford Dictionary. The focus in this work is on the designing of a new and efficient Lemmatizer which is more competent than the existing freely available lemmatizers. The analysis of the detailed literature survey conducted during this study as well as the model developed has been published in UGC Care journal and Springer Publication.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1. Introduction

This chapter discusses the morphological structure of English words which eventually leads to the designing and developing of the proposed model. It also covers an introduction to the stemming, lemmatization and morphological analysis techniques. In this chapter, problem formulation, problem statement, objective and research contribution are also deliberated.

1.1 Morphology

"The term morphology is generally attributed to the German poet, novelist, playwright, and philosopher Johann Wolfgang von Goethe (1749– 1832), who coined it early in the nineteenth century in a biological context. Its etymology is Greek: morph means 'shape, form', and morphology is the study of form or forms. In biology, morphology refers to the study of the form and structure of organisms. In linguistics morphology refers to the mental system involved in word formation or to the branch of linguistics that deals with words, their internal structure, and how they are formed." Mark Aronoff [5] .

Morpheme is described as the smallest meaningful linguistic pieces with a grammatical function. A morpheme is not necessarily the same as a word. E.g. the "happy" and "-ness" of "happiness" are identified as morphemes, which cannot be further divided into smaller meaningful parts. Now, "consideration" is another word in which there are two morphemes: "consider" and "–ation". [5]

A lexeme is a unit of morphological analysis which represents a set of morphed words in linguistics. Two types of lexemes are available for the formation of word structure: inflection and derivation in English grammar. Inflection involves the formation of grammatical forms: past, present, future; singular, plural; masculine, feminine and neuter. The use of these grammatical forms is generally ruled by sentence structure. In English, regular verb lexemes have a lexical stem, which is the bare form with no affixes (e.g., jump) and three more inflected forms, one each with the suffixes -s, -ed, and -ing (jump, jumped, and jumping). Noun lexemes have a singular and plural form. Adjectives, adverbs, prepositions, and other parts of speech typically

exist in a single form in English. [5]

Lexeme in derivation involves the creation of a new lexeme from an existing lexeme, such as “employer” or “employment” from “employ”. In many cases, the Part-of-speech (POS) of a new lexeme has been changed into different POS’s class after attaching a derivational suffix. E.g. Verbs to nouns: employ + er; Verb to Adjective: accept + able; Nouns to nouns: fish + ery; India + ian; Adjectives to adjectives: blue + ish; e.g. Derivative function creates agent nouns from verbs : $X]_V$ $er]_N$; e.g.: $think]_V$ $er]_N$, $run\underline{n}]_V$ $er]_N$ etc.[5]

So, this proposed lemmatizer (LemmaChase and LemmaQuest) has been mainly designed to extract base lexeme from derivational lexeme. In dictionary, there is no logical or syntactical mapping between derivative lexeme and root lexeme; both exist as independent words in any dictionary. This model is composed of a number of phases, wherein after the execution of each phase, the derivative lexeme or nominalized word comes closer to its lemma. It has been designed by considering the part-of-speech of each word in the given context of a text, which is being pre-processed by Stanford POS tagger tool.

In stemming and lemmatization processes, two term conflation approaches: non-linguistic and linguistic are basically used. Most of the Stemming algorithms are developed based on linguistic studies and on word morphology; they do not utilize methods pertaining to NLP. Figure 1.1 depicts a concise categorization of term conflation methods. [6]

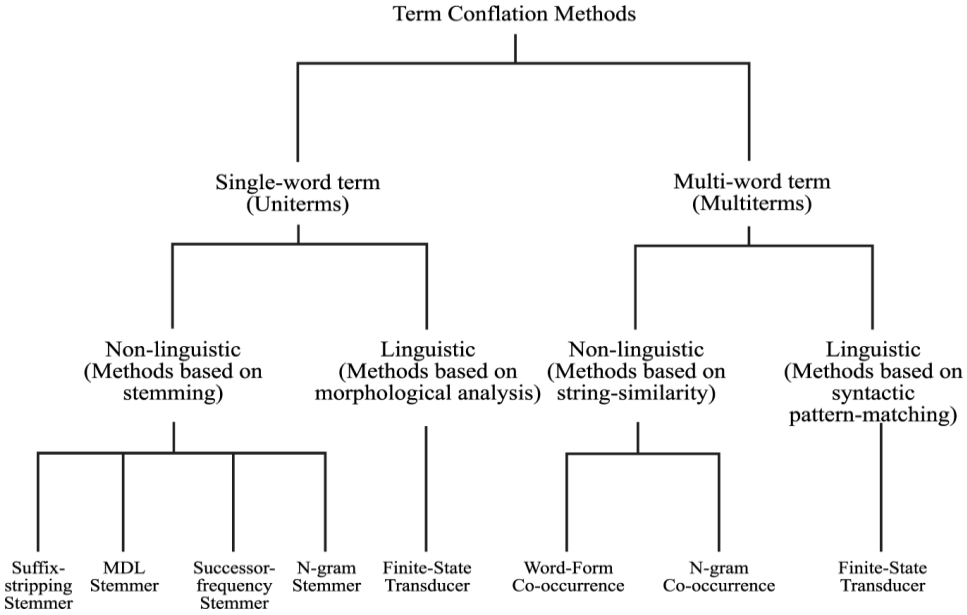

Fig. 1.1 Category of Term Conflation Methods

Now, in this research work, "Term-Conflation-Methods" are applied only on single-word term (Uniterms). Many stemming algorithms generate good results for the conflation and normalization of uni-term variants (Porter, 1980; Frakes, 1992). Within suffix-stripping stemmer, the most effective technique is the **longest match algorithm** which was applied in Lovins' stemming procedure. In stemming, the morpheme left after affix elimination can hardly be used for IR and IE purposes because most of the time, that morpheme may not be a dictionary word. The solution to this problem resides in doing a full morphological analysis, and this task can only be processed by Lemmatizer. [6]

In an article, Goldsmith (2001) claimed that automatic morphological analysis can be divided into four major approaches. The first approach, based on the work explored by Harris (1955) and further designed by Hafer and Weiss (1974), provides a goodness of break between letters that can be compute by the successor frequency there, compared to the successor frequency of the letters on either side [6]. The second approach looks for n-grams, which are likely to be morpheme-internal. The third approach highlights the discovery of patterns of phonological and morphological relationships between pairs of words: a base form and an inflected form. The fourth approach explores unsupervised learning techniques, yielding a partition of stems and affixes, segmenting longer strings into smaller units (Kazakov, 1997; Kazakov and Manandhar, 2001). [6]

(1) **Non-linguistic Approach**: "**Stemming** methods consist mainly of suffix stripping, stem-suffix segmentation rules, similarity measures and clustering techniques." [6]

(2) **Linguistic Approach**: "**Lemmatization** methods consist of morphological analysis. That is, term conflation based on the regular relations, or equivalence relations, between inflectional forms and canonical forms, represented in finite-state transducers (FST)." [6]

Some studies have uncovered that indexing by the stem does not substantially improve the performance of retrieval, at least not in the English language (Harman, 1991) [1,4, 6]. Researcher, Harman, observed that the use of a stemmer in the query is intuitive to many users, and reduces the number of terms decreasing the size of the index files, but retrieves too many non-relevant documents. This problem can be minimized by the process of lemmatization [6]. The proposed lemmatization model is developed based on both stemming and lemmatization techniques.

1.2 Approaches to Stemming

W.B Frake, renowned IR researcher**:** Software Engineering Guild, Sterling, categorized the stemming process into four major techniques as follows: [1, 2]

1. Affix-based or Rule-based
2. Statistical or Successor Variety
3. Table lookup
4. N-gram Statistical Approach.

Then these four categories were further extended to a "mixed" approach. Based on mixed approach; Corpus-Based, Context-sensitive, K-stemming and Xerox stemming were developed. Some other statistical techniques YASS, HMM were also incorporated into language-independent stemming methods in later stage. Fig. 1.2 [8] depicts the different categories of stemming algorithms. Under Affix-based/ Rule-based stemming, sub-categories are as follows: i) Porter, ii) Lovins, iii) Paise & Husk, iv) Dawson. [8]

Under statistical approach, different stemming techniques are as follows:

i) Successor Variety, ii) HMM, iii) YASS and iv) N-gram.

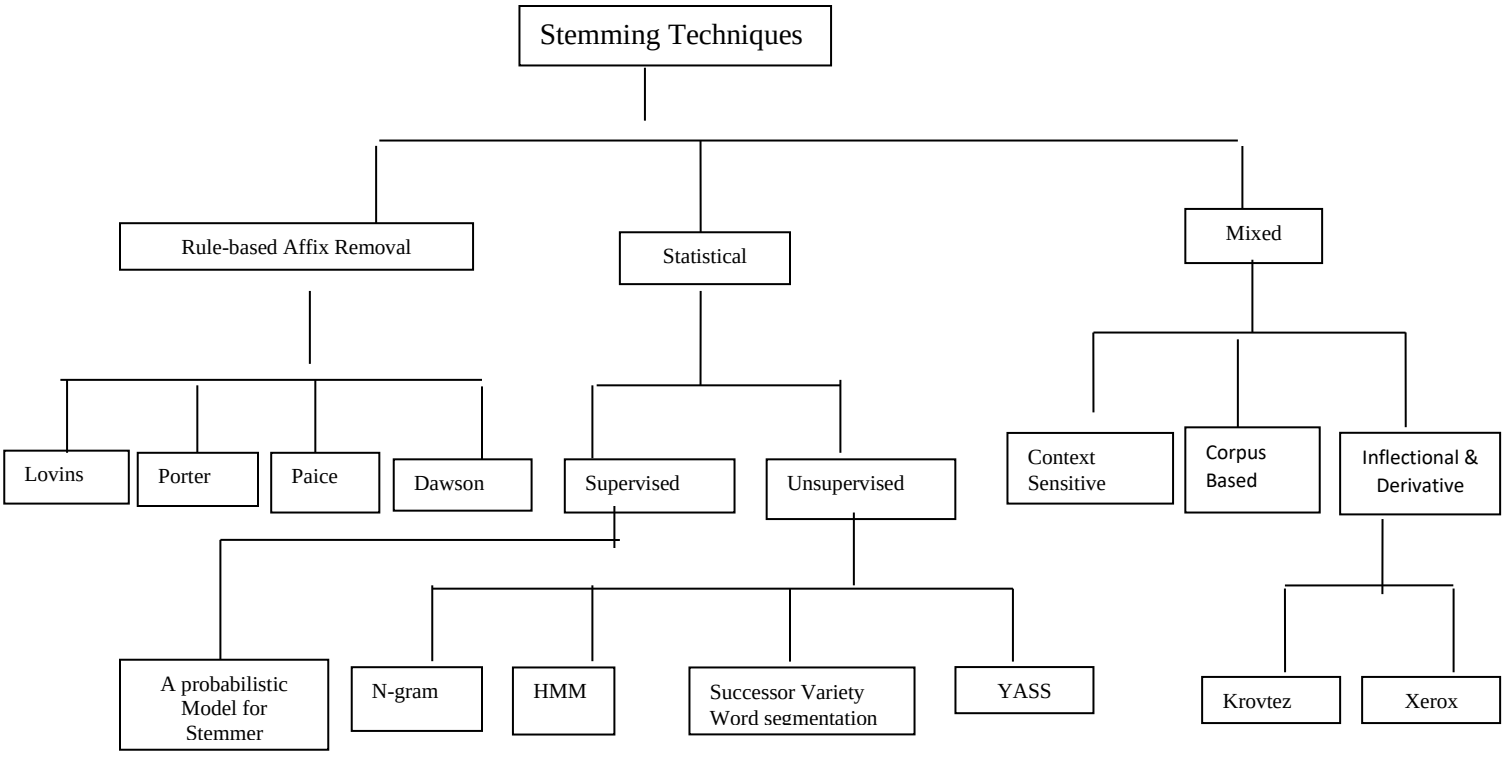

Fig. 1.2 Stemming Category

Rule-based or truncating stemmers apply a set of English morphological pre-defined conditional and transformation rules on textual words to generate a single token from different

inflectional and derivational variants after chopping off suffixes and prefixes without exploiting linguistic basis (e.g. schools to school/woman to wom/believe to belief/central, center to cent).

First one was Professor John W.Tukey's stemming algorithm at Princeton University, second attempt was Michael Lesk's algorithm, developed at Harvard University, under the direction of Professor Gerard Salton, and a third algorithm was developed by James L. Dolby of R and D Consultants, Los Altos, California. [10]

Dawson Stemmer (1974) is an extension of the Lovins approach except that it covers a much more comprehensive list of about 1200 suffixes. Like Lovins, it is a single-pass stemmer and hence is pretty fast. The suffixes are stored in the reversed order indexed by their length and last letter in the Dawson algorithm. [3, 8]

In 1980, **Martin Porter** developed a stemmer which is also widely used in the area of text mining and NLP. [17, 18]

In late 1980, an iterative **Paice-Husk stemming** algorithm was designed and developed by Chris D Paice at Lancaster University. This algorithm incorporated a compact set of stemming rules, specifying the removal or replacement of an ending of a word. Paice also developed a direct measurement for comparing stemmers based on counting the over-stemming and under-stemming errors. [7, 19, 20]

The **statistical stemming** is developed based on a novel string distance measure which leads to lexicon clustering. The results show that **stemming** significantly improves retrieval performance (as expected) by about 9-10%, and the performance of the **statistical** stemmer is comparable with that of the rule-based stemmer. In the absence of extensive linguistic resources for certain languages, statistical language processing tools have been successfully used to improve the performance of IR systems and of the IE system. Rule-based stemmers for resource-poor languages are either unavailable or lack comprehensive coverage, in all those cases, a statistical approach can successfully stem any word for that language [11]. Two types of statistical approaches are applied to develop different stemmers, one approach is unsupervised and another one is supervised. [12]

In an **unsupervised** approach, stemming aims to identify morphological classes that share common roots based on different string similarity distance. This type of stemming is elaborately discussed in literature review section [13]. Stemming model based on **supervised approach**, describes the mutual reinforcement relationship between stems and derivations and

then provides a probabilistic interpretation. Stemming algorithms based on statistical methods ensure no additional costs to add new languages to the system. This is an advantage that becomes crucial, especially for applications like digital libraries which are often constructed for a particular institution or nation, and are able to manage a great number of non-English documents as well as documents written in many different languages.

N-gram token generation, statistical based stemmer was also developed by George W. Adamson and Jillian Boreham in 1974. Though it is called "stemming method", but a stem token is not produced in this algorithm. This method has no requirement of grammatical knowledge of the respective natural language. It is elaborately explained in the literature review chapter. [14]

Statistical based approach, "Successor Variety Word segmentation"**,** was introduced by Marget A. Hafer & Stephen F. Weiss in 1971 and this stemmer had aimed to generate segment from word based on structural linguistic [15]. Successor Variety decomposes a word across its' structure to generate the segment. This process is also elaborately discussed in literature review chapter. **Benno Stein and Martin Potthast** also introduced statistical stemming concept based on Successor variety approach named "**Successor Variety for Stemming**" in 2007. [16]

Massimo Melucci and Nicola Orioe developed a stemming approach based on statistical **HMM** concept by publishing paper "**Algorithm for Probabilistic Stemming"** in 2005. [13]

Prasenjit Majumder, Mandar Mitra, Swapan K. Parui, and Govinda Kole, ISI (2007) have proposed a stemming algorithm, independent of linguistic expertise. This Stemming [**YASS**] has a target to identify morphological classes that share common roots based on some statistical string similarity measures which is implemented in the proposed "LemmaQuest" algorithm. YASS algorithm is basically developed based on unsupervised statistical approach [12].

Michela Bacchin, Nicola Ferro and Massimo Melucci (2005) developed a language-independent probabilistic stemmer. This model is applied into several languages. It describes the mutual reinforcement relationship between stems and derivations and then provides a probabilistic interpretation. This model splits each word of finite collection of words into all possible positions and then creates suffix list. The stemmer accepts word as an input and it tries to determine the split which corresponds to a stem and a derivation. The stemmer uses the linguistic knowledge. It also considers only the pairs which lead to the word and not the whole collection [67].

Jinxi Xu and W. Bruce Croft developed "**Corpus-Based Stemming using Co-occurrence** of Word" around year 1994/1995 under mixed category of stemming and they also followed statistical approach [21]. They adapted to language characteristics of a domain as reflected in a corpus. e.g. "stocks" is plural form of "stock" in Wall Street Journal (WSJ), but primary meaning of "stocks" in corpus related with **Medieval history** are the devices to punish a prisoner for public humiliation [21]. So, a good conflation for one corpus may be bad performer for another. Corpus-based statistics may guide algorithm to establish relationship between any two corpus words. Corpus-based stemmer helps to prevent to conflate "policy / police", "addition/ additive", university/universal" all these un-related words into single stem token. It is observed that such semantically un-related words rarely co-exist in a corpus. This concept is applied here through statistical approach. Here, affix removal stemmer's fails and wrongly conflate all above mentioned word pairs. Corpus-Based stemming process is as follows: Assumptions are: Word variants that should be conflated will occur in same documents or in some text window (100 words text window). [21]

Following metric to measure the significance of "**word form co-occurrence"**

em(a,b)= max((n_{ab} − En(a, b))/(n_a + n_b)), 0) where En(a, b) = k$n_a n_b$ is the expected number of co-occurrences assuming a and b are statistically independent where n_a, n_b are the number of occurrence of words a and b in the corpus. $n_a n_b$ is the number of times both a and b fall in given text window or a single document. K is the parameter which is estimated for corpus [21]. This metric is a variation of EMIM (**Expected Mutual Information Measure**) [Church and Hanks 1989; Van Rijsbergen 1979]. It is used to measure **significance of association**. For word form co-occurrences, EMIM can be defined as EMIM (a,b) = P(a,b)$\log_{10}$(P(a,b)/P(a)P(b)) where **P(a,b)= n_{ab}/N**, [21]. **P(a)=n_a/N, P(b)=n_b/N,** N is the number of text windows "the" & "of" , "new" & "news" and "gas" & "gases". "Gas" indícate as "natural gas"/ HydroCarbon gas where as "gases" indícate "inert gas"/ "hot gas"/ "helium", "neon", "xenón" [21]. "**News**" and "**new**" co-occur frequently, but they are un-related word pair. According to formula for estimating En(a,b), their expected co-occurrence is 50,000. but **em score** is 0, which suggests that they are not related. **em scores** of sample word pairs on WSJ with 100 words window [21].

Jiaul H. Paik, Swapan K. Parui (ISI, Kolkata) published a paper "**A Novel Corpus-Based Stemming Algorithm using Co-occurrence Statistics**" in 2011. [23]

Context sensitive Stemming is a method where stemming is done before indexing a document. Context sensitive analysis is done using statistical modeling on the query side [22]. This method was proposed by **Fuchun Peng, Nawaaz Ahmed, Xin Li and Yumao Lu**. This stemming process is divided into four steps after the query is triggered. Steps: Candidate generation, Query Segmentation and head word detection are implemented in this approach.

The **Krovetz stemmer** (KStem) was developed in 1993 by Robert Krovetz and is a linguistic lexical validation stemmer. It effectively and accurately removes inflectional suffixes. The algorithm of KStem is described in the next chapter [24].

Observation from existing Stemming Algorithms: It was noticed that there is a lot of similarities between the stemming algorithms, and if one algorithm scores better in one area, the other does better in some other area. In fact, none of them give 100% output but are good enough to be applied to the TM, NLP or IR applications. The main difference lies in using either a rule-based approach or a linguistic one. A rule-based approach may not always give the correct output and the stems generated may not always be correct words. As far as the linguistic approach is concerned, since these methods are based on a lexicon, words outside the lexicon are not stemmed properly. A statistical stemmer may be language independent but does not always give a reliable and correct stem.

1.3 Lemmatization

Unlike stemming, where a lot of work has been done and, many stemmers have been popularized, there is still scope in designing of a lemmatizer as discussed before. It has been observed from literature survey that broadly seven different categorical approaches have been used in designing of lemmatizers. Below-mentioned figure 1.3 shows broad categorization of Lemmatization techniques. The detail functionality of all above mentioned Lemmatization approaches are discussed in the "Literature Review" chapter.

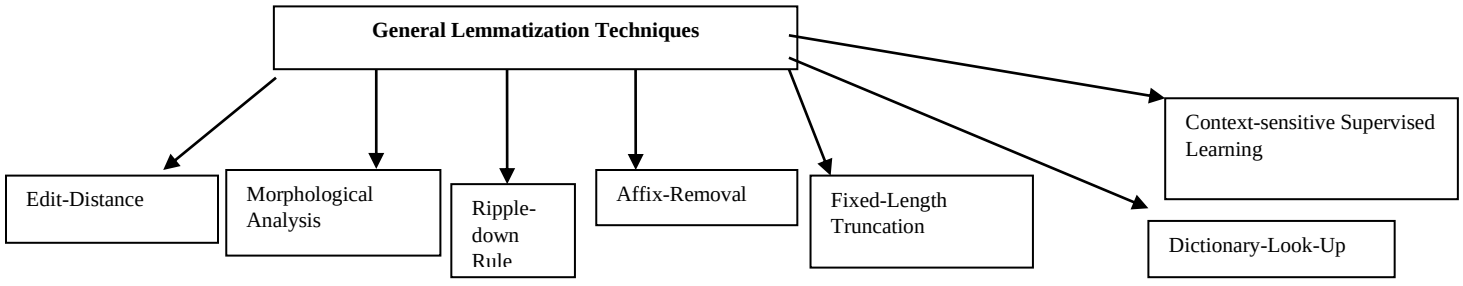

Fig.1.3 Lemmatization Techniques

In any Lemmatization process, the following set of input is given and a set of below-mentioned output is expected to display.

Input (Source Text)

- Input: Text file and dictionary morphed-word list with short or long in length.
- .Effective text: collection of words including allied inflected, derivative and nominalized words.
- Dictionary: collection of the maximum number of morphed and derivative words.

Purpose

- Use: the Output file generated by lemmatizer, is acted as an input file for IE, IR and a NLP tool.
- Audience: Text miner, web searcher, natural language processor.

Output (Generation of Tokens)

- Generation of Tokens: List of root words (number of tokens : m), are meaningful dictionary words for unique input-words (number of tokens: n). m<<n
- Generation of a comparative grid: List of root words generated by the proposed lemmatizer and by existing popular lemmatizers.

1.4 History of Morphological Analysis

The text, written in natural language was started to analyze automatically from 1954. Detail historical description is listed in table 1.1.

Table 1.1 History of IE, IR and NLP

IE:

1954	The first automatic translations tool had been developed from English to the Russian language in 1954, but automation was limited to a handful of sentences.[69]
late 1970s	Automation in "information Extraction" had been initiated back to the NLP.[69]
Mid-1980	**JASPER** tool had been popularized for Reuters to provide real-time financial news.[69]
Beginning in 1987, 1989	IE was spurred by a series of Message Understanding Conferences. MUC is a competition-based conference that focused on the following domains: MUC-1 (1987), MUC-2 (1989): Naval operations messages.[69]
1998	MUC-7: Satellite launches reports.[69]

IR:

1970s	First online systems—NLM's AIM-TWX, MEDLINE; Lockheed's Dialog; SDC's ORBIT.
1975	Three highly influential publications by Salton fully articulated his vector processing framework and

	term discrimination model. A Theory of Term Importance in Automatic Text Analysis (JASIS v. 26) [70]
1979	C. J. van Rijsbergen published Information Retrieval (Butterworths). Heavy emphasis on probabilistic models. [70]
1989	First World Wide Web proposals by Tim Berners-Lee at CERN. [70]
1999	Publication of Ricardo Baeza-Yates and Berthier Ribeiro-Neto's Modern Information Retrieval by Addison Wesley, the first book that attempts to cover all IR. [70]

NLP:

1960	Some notably successful natural language processing systems were developed in the 1960s were SHRDLU. [71]
1966	ELIZA, a simulation of a Rogerian psychotherapist, written by Joseph Weizenbaum was created.
1990	The research on the core topics such as word sense disambiguation and statistically colored NLP got a direction of research. [71]
1990s - 2010	**Statistical NLP**: Up to the 1980s, most NLP systems were based on complex sets of hand-written rules. Starting in the late 1980s, there was a revolution in NLP with the introduction of machine learning algorithms for language processing. With the growth of the web, research has thus increasingly focused on unsupervised and semi-supervised learning algorithms. e.g., text and speech processing, text-to-speech processing, word segmentation (tokenization), lemmatization and stemming, morphological segmentation and part-of-speech tagging. [71]

1.5 Applications of Stemming and Lemmatization

Most useful applications of IE, IR and NLP cannot be developed without the implementation of stemming or lemmatization. Some are described below.

i) Text-Summarization: In Automatic Text-Summarization, preprocessing is an important phase to reduce the space of textual representation. Classically, stemming and lemmatization have been widely used for normalizing words. However, even using normalization on large texts, the curse of dimensionality can disturb the performance of summarizers [26]. Morphological analysis is a very important phase of pre-processing of NLP systems because it allows reducing the dimension of the vector space representation in an IR system [27, 28]. Several other applications such as Document Indexing, Textual Classification and Question-Answering systems among others utilize this reduction.

ii) Application of NLP: Regardless of our text data format, steps that are used to solve NLP problems remain more or less same. Major steps are depicted in Fig 1.4, while solving the NLP problems.

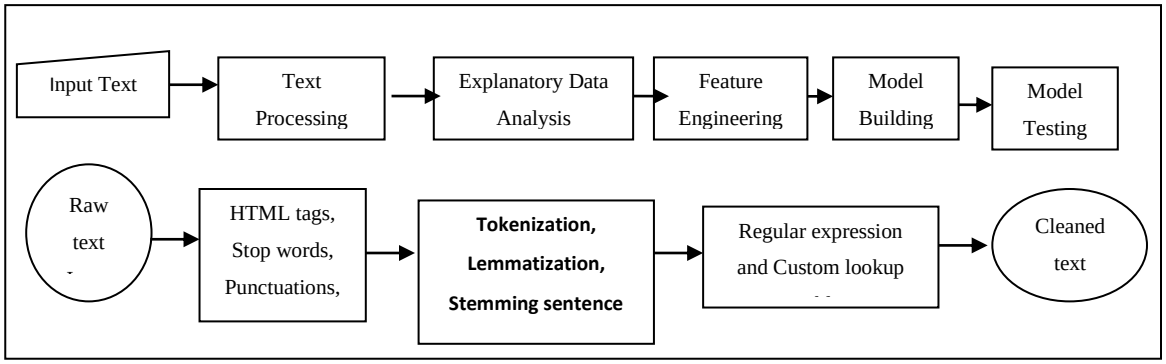

Fig 1.4 Steps in NLP

iii) Text Normalization/Text Simplification: On a higher level, normalization is used to reduce the dimensions of the features, so some machine learning models can efficiently process the data. Text data contains multiple representations of the same word which will be merged into a single word through lemmatization. e.g., "player", "played", "plays" and "playing" are different variations of the word "play".

1.6 Problem Formulation

One of the important pre-requisite tasks of TM, IE, IR and NLP applications like Name Entity Recognition, Summarization, Text Simplification, Semantic Analysis, etc. is to trim the text into number of significant base words through which sentiment of the text would be accurately mined. For extracting a limited collection of dictionary base-words from all morphed and derived words available in the given text, lemmatization process is more effective as compared to stemming process because stemming processes are not able to generate dictionary base-words most of the time.

It has been observed that the existing popular lemmatizers are not still handling the derived morphed words and nominalized words like application, applicant, judgmental, employer, employee, safely, etc., in the sense that the lemmas of these words are not correctly generated. They can only handle inflected words like women, running, broken, applying, applied, employs, etc. to generate their lemmas. Some of the stemming processers can properly handle both inflected and derived words and can generate single stem for all morphed allied words e.g. the Porter's Stemmer generates single stem 'applic' for the words – applicant, application and 'accept' for the words – acceptance, acceptable. However, most of the time, all these stems are not necessarily dictionary words like the stem 'applic' which is not a real word.

Like stemmers, most of the popular lemmatizers are also not able to generate the correct lemmas for all English derived words. Therefore only stemming or only lemmatization process cannot provide us a proper solution in generating the dictionary root words for both inflected and derived English words. This limitation can affect the NLP, TM, IE, etc. applications and their output drastically. There is a need to therefore develop a lemmatizer which would handle this drawback and generate a better pre-processing output.

A possible solution would be to generate stem token through suitable stemming process and then to construct a meaningful dictionary word from stem token through proper morphological analysis. The existing stemmers and lemmatizers generate the stems or lemmas respectively after processing each individual input word in the given input text. For each word a number of steps are to be performed. A simpler and efficient model has been proposed to minimize the number of steps which creates clusters for allied morphed words through language independent statistical computations. For each independent cluster, a single lemma can be generated through a single pass, irrespective of number of allied words present in that cluster. The work has been done for English morphed words only.

1.6 Problem Statement

The aim of this study is to design, develop and implement a Lemmatizer for English morphed words handling nominalization and giving a better performance and output as compared to the existing popular lemmatizers.

1.7 Objectives of the Study

The objective of this research work is to develop a simple, robust and enhanced model for lemmatization. This model is proposed to overcome the shortcoming of the existing stemmers and lemmatizers, generating an accurate, precise and exact output in terms of lemmas for the input text. This intended model is anticipated to serve the following purpose and objectives:

1. To study and to implement existing stemming and lemmatization approaches and to compare the limitations and advantages of each.
2. To incorporate grammatical word formation rules for constructing English derivative words and to apply statistical distance measures to minimize the erroneous result and limitations of the existing stemmers and lemmatizers.

3. To design a lemmatizer which generates correct lemmas for different morphed, derived and nominalized words and comparing them with the standard and popular lemmatizers. The lemmatizer should work on the text as a whole combining related morphed /derived words making the processing and output simpler and acceptable.

1.8 Research Contribution

Two models are presented and implemented that offer a generic and logical solution to meet the research objectives.

The first proposed model-LemmaChase provides a morphological analyzer in which word formation rules have been incorporated to generate a new dictionary word from a derived input word. The list of lemmas which are extracted for the input are more accurate and error-free as compared to the existing available lemmatizers. It also handles inflected input words properly just like other lemmatizers.

The second proposed model-LemmaQuest is very capable and efficient when the input text has a large number of allied morphed words which coexist in that text. This lemmatizer works on the whole input text instead of executing on each derived or morphed word separately. This model generates groups of allied words dynamically to minimize the lemma extraction processing task. It can generate a single lemma for a group instead of generating a lemma for an individual input word. The model segments words to generate stem-token which is further processed to generate the correct lemma.

After the execution of LemmaQuest, it is observed that the output generated by it is far more accurate and precise, and the error rate is far less as compared to the output by other existing lemmatizers.

1.9 Conclusion and Summary

In this chapter, the necessity to design and develop these two proposed models is explained through exploring existing stemming, lemmatization and morphological analysis in linguistics. Impact of execution of these two models on IE, IR, and NLP is also identified.

Chapter 2
Literature Survey

2. Overview of Literature Survey

The proposed models are the combination of stemming and lemmatization techniques. So detail study of existing stemming and lemmatization algorithms is required to design a new lemmatizer. Affix-removal Lovins and Paice stemmers' techniques [9, 19] are incorporated to design proposed lemmatization models (LemmaChase and LemmaQuest) after studying all existing stemmers and after observation of the result of all Affix-removal stemmers. Available Affix-removal stemmers, statistical stemmers and lemmatization approaches are elaborately discussed in this chapter with continuation of the previous chapter's discussion.

2.1 Analysis of Affix Removal Stemming Algorithms

Lovins' Stemmer:

Lovins' stemming algorithm is developed based on two major concepts: iteration and longest-match. Iteration is handling suffixes of any morphed English word to generate a dictionary root word or stem token**.** It uses 294 endings, 29 conditions and 35 transformation rules to generate stem token form different morphed word variants. Lovins' algorithm handles inflectional suffices as well as irregular singular-plural suffices. Below mentioned steps broadly describes Lovins' stemming algorithm [9].

1. The algorithm starts with "Longest–match" technique. It determines the position from where an ending list begins searching for a match, so that the stem is at least two characters long and then searches the ending list for a match to the last part of the word being stemmed (e.g., metallic to metall, metal to met, metallically to metall, magnetizable to magnet). If the longest ending is found with its associated given condition and, then it is removed.
2. If an ending suffix is found from the given 294 suffices ending list, that suffix is checked as to whether the pre-defined context sensitive rule is satisfied or not (e.g., induction to induct). If that word's contextual suffix is satisfying the condition, then the suffix will be

removed to generate the 1st phase stem. For e.g., for the word "believed", context sensitive rule "ed E" is found and then condition rule "E" is applied to generate "believ".

3. Now, the 1st phase stem will be re-coded for generating the final one. The final consonant will be un-doubled, if exists within the stem. Stem is recoded based on a matched transformation rule retrieved from 35 word endings transformation rules, and then the final stem token will be created (e.g., induct to induc, metal to metal, magnet to magnet). If no transformation rule is matched, previously generated stem will be declared as the final stem.

Table 2.1 Sample List of Suffices and Recoding Rules of Lovins [9]

Appendix A. The list of endings	Codes for context-sensitive rules associated with certain endings	Appendix C. Transformation rules used in recoding stem terminations
alistically B, arizability aizationally B, izationally B	A: No restrictions on stem B: Minimum stem length = 3	iev -> ief rpt -> rb ix -> ic
ability A, aically A, alistic balities A, es E, ionality A, ionalize A, iousness A, izations A	C:Minimum stem length = 4 D:Minimum stem length = 5	ax -> ac ex -> ec ix -> ic

Table 2.2 Sample Set of Output of Lovins Stemmer

Word	Stemmed Term				Word	Stemmed Term			
	Porter	Lovins	Paice	KStem		Porter	Lovins	Paice	KStem
Believe	believ	belief	Believ	Believe	Index	index	indic	index	index
Belief	belief	belief	Believ	Belief	Indices	indic	indic	Ind	indices
Running	Run	run	Run	Running	Formulae	formula	forml	formula	formulae
Run	Run	run	Run	Run	Formula	formula	forml	formul	formula

Porter's Stemmer:

This stemmer checks the presence of suffices within a word and then matches that suffix with already stored suffices. The algorithm's six steps are listed below [17, 18]:

This step identifies suffices and recodes them accordingly.

1.1 It identifies and removes suffix ("-sses", "-ies", "-s") from words having plural part-of-speech and recodes to generate stem token (e.g. actresses to actress, abilities to ability, ponies to pony & pennies to penni).

1.2 It also removes ending "–ed" used in the past tense and ending "-ing" used in the present continuous tense from words and recoded them. (e.g., abbreviated to abbreviate & abbreviate to abbreviate) and then, applying recode rule to convert "abbreviat" into "abbreviate".

2. The second step recodes terminal 'y' into 'i' in presence of another vowel within a stem or original word. e.g.," penny" to "penni", "ability" to "ability".
3. Third step eliminates double suffices and also does recoding based on a given list of suffices. E.g., from "running", -"ing" suffix is removed and then "runn" is recoded into "run".
   3.1 It indexes the 2nd last character of the stem token. It maps double suffices to a single suffix.
   3.2 This step also recodes "-ational", "-tional" into "-ate", "-tion" respectively. E.g.,"application" into "applicate". e.g., "applicate" to "applic" using recoding rule "- icate>"-ic").
4. It eliminates "-ance", "-er" endings from stem words using <c>vcvc<v>. e.g. "acceptance" and "acceptable" into "accept".
5. Final step removes "-e" ending from the input stem which is received from the 5th step. e.g., "assemble" to "assembl".

In this algorithm, a word "generalizations" is stemmed into a word "generalization" in step-1 and, then into a word "generalize" in step-2. Step-3 converts it into a word "general" and, then the final stem "gener" will be developed by step-5.

Table 2.3 Sample Set of Output of Porter Stemmer

Word	Stemmed terms			Word	Stemmed terms		
	Porter	Lovins	Paice		Porter	Lovins	Paice
Reporters	report	reporter	Report	Applies	Appli	appl	apply
Reporting	Report	report	Report	Apply	Appli	ap	apply
Acceptance	Accept	accept	acceiv	Application	Applic	applic	apply
Acceptable	Accept	accept	acceiv	Abilities	Abil	abil	abl
Applicant	Applic	appl	appl	Ability	Abil	abil	abl

Paice Stemmer:

Paice algorithm is working based on a rule table [19]. This stemmer has quite similarity with Lovin's approach. Each rule directs either for deletion or for replacement of endings from a word to generate a stem token. The rules are clustered into sections corresponding to the final letter of the suffix. The rule table is searched quickly by looking up the final letter of the current word or truncated word. Within each section, the ordering of rules plays a meaningful role. Each rule comprises five elements, two of which are optional:

a) " An ending of one or more characters, held in reverse order;" [19]

b) ” An optional intact flag "*";”[19]

c) ” A digit specifying the remove total (may be zero);” [19]

d) ” An optional append string of one or more characters;” [19]

e) ” A continuation symbol, ">" or "."” [19]

Paice stemming algorithm is given below:

1. “Selection of relevant section: First, the final letter of the form is inspected (For e.g. “center” has final letter ‘r’) ; if there is no section corresponding to that letter, then process will be terminated; otherwise, first rule (“re2 {-er>-}”) in the relevant section is applied (for “center”/ “hsiug5ct” {-guish>-ct}” for “distinguish”).” [19]
2. “Check applicability of rule: If the final letters of the form are not matched with the reversed ending string in the rule, then process will jump into step 4; if the ending is matched, and the intact flag is set, and the form is not intact, then process will jump into 4; if the acceptability conditions are not satisfied, then process will move to 4.”
3. “Apply rule: The number of characters specified in the “remove total” (mentioned above in rule elements) are deleted from right hand of the form; if there is an “append string”, then it is appended to the form; (e.g. “-er” is removed from “center”/ “-guish” is removed from “distinguish”). If the continuation symbol "." exists into rule element, then the process will be terminated.”
4. ” Look for another rule: Process will move to the next rule in the table; if the section letter has been changed, then process will be terminated; otherwise process will move to 2 (ai*2. {-ia > - if intact}; a*1.{ -a > - if intact }).”

Table2.4 Sample Set of Output of Paice Stemmer

Word	Stemmed terms					Stemmed terms				
	Porter	Lovins	Paice	KStem		Word	Porter	Lovins	Paice	KStem
Center	center	Center	cent	center		Ox	ox	ox	Ox	ox
Central	central	Centr	cent	central		Oxen	oxen	oxen	Ox	oxen
Woman	woman	Woman	wom	woman		Distinguishing	distinguish	distinguish	distinct	distinguish
Women	women	Wom	wom	women		Distinguish	distinguish	distingue	distinct	distinguish

Krovetz-Stemmer Stemming (KStem):

The primary task of KStem is already described in the “Introduction” section. Now, the algorithm is explained in the below-mentioned steps [8, 24]:

1. Converting the plurals of a word to its singular form. [8]
2. Transforming the past tense of a word to its present tense. [8]
3. Truncating the suffix ‘ing’. [8]

This algorithm uses a machine-readable dictionary. It uses inflectional affix removal and transformation rule as well as already stored dictionary. The broad steps of this **KStem** algorithm are described below [24].

1. A list of derivational endings from Longman dictionary and different term dictionaries are initially created containing some root words and derivational morphed words. exceptionWords: {"bathe", "caste", "cute"}, directConflations:{"aging", "age"}, {"dying", "die“}}. Different dictionary words (a to z) are stored within 8 different files.
2. The word is either plural or not, is checked with the help of a plural inflectional rule function after passing through an exceptional word list.
3. Then, trimmed stem or original word is searched in a dictionary, if it exists, its root word is returned, otherwise the word will be passed through pastTense inflectional rule handler function.
4. The word in “verb” or “noun” part-of-speech form, ended with “-ing”, is stemmed in aspect handler function. But, function handles short words (aging -> age) via a direct mapping. This prevents (thing -> the) “thing” from being retrieved without stemming via dictionary. If the input document size is large, this stemmer will become weak at performance. This stemmer does not consistently produce a good recall and precision performance.

Table 2.5 Sample Output of KStem Stemmer

Word	Stemmed terms				Stemmed terms				
	Porter	Lovins	Paice	KStem	Word	Porter	Lovins	Paice	KStem
Authority	author	author	auth	Authority	State	state	st	stat	state
Author	author	author	auth	Author	statement	statement	stat	stat	statement
Authorize	author	author	auth	Authorize	Station	station	stat	stat	station
Factory	factori	factor	fact	Factory	News	new	new	new	news
Factor	factor	fact	fact	Factor	New	new	new	new	new
Factorial	factori	factor	fact	Factorial	Age	ag	ag	age	age
factorize	factor	factor	fact	Factorize	Aging	ag	aging	ag	age

Table 2.6 Summarization of Existing Popular Stemming Algorithms [8]

Stemmer	Method	External Resource	Stem/ Lemma	Application	Advantage	Disadvantage
Lovins, Paice, Porter	Affix Removal	Affix removal & transformation rule	Stem token	Information Retrieval/Extraction	Fast-single Pass Algo., Handle many irregular plurals.	Not very Reliable and fails to form words from stem in many times.
KStem	Dictionary LookUp	Affix removal & transformation rule, stored dictionary.	Lemma	Information Retrieval/Extraction	Generate dictionary word instead of invalid stem most of the times	Cannot consistently produce a good recall and precision
N-gram	Statistical	NIL	Di-gram	Clustering	Language Independent	Require significant amount of space for creating and indexing n-gram
Successor Variety	Statistical	Threshold value	Stem token	Language independent extraction/ retrieval	Language Independent	
Corpus-Based	Statistical	EM, window size	Word grouping	Clustering/ grouping	Appropriate for a given corpus, Valid word is generated after stemming	Need to develop individual statistical measure for every corpus separately.
Context-sensitive	Head word detection, Document matching	Document	Lemma	Web search	Improves selective word expansion on the query side.	High processing time, complex algorithm

2.2 Introduction to Lemmatization

Lemmatization techniques are also precisely discussed in the “Introduction’ section. The proposed model “LemmaChase” has targeted nominalization words for generating the correct lemma for them. These nominalized words are being treated as separate independent words in the English dictionary without connecting them to their actual root words or lemmas. A derivational affix is added for a word (Verb) to convert it to a noun form. e.g., the noun “legalization” has been produced from the verb “legal”. These word formations are also called “nouning”. This is a challenge to connect nominalized word with its base word (lemma). So, LemmaChase has tried to meet this challenge. Examples of Nouns formed from adjectives are “applicability” (from “applicable”), “carelessness” (from “careless”), “difficulty” (from “difficult”) and “intensity” (from “intense”). Examples of Nouns formed from verbs are “failure” (from “fail”), “nominalization” (from “nominalize”), and “investigation” (from investigate”). The nominalized word shows only a single Part of Speech (POS) in any English dictionary, though its root word basically exists in different POS forms and the root word’s morphological structure also differs from nominalized word (e.g, “judgmental” to “judge”/ “application” to “apply”/ “angrily to angry”).

2.3 Analysis of Lemmatizers

Unlike stemming, where a lot of work has been done and many stemmers have been popularized, there is still scope in designing a better performed lemmatizer as discussed before.

First Approach of Lemmatization:

In lemmatization, the first approach, "**Edit Distance**" (also known as Levenshtein Distance) has been implemented on a dictionary-based algorithm in order to get the automatic induction of the normalized form (lemma) of regular and blandly irregular words without direct supervision [53]. The algorithm is a composition of two alignment approaches based on the string similarity and the presence of the most frequent inflexional suffixes. In the "Edit Distance" algorithm, Levenshtein distances in-between strings are stored in a simple metric which can be used as an effective string approximation tool.

The Edit Distance is calculated based on a simple dynamic programming algorithm which focuses on a primitive string edit operation [53]. This operation is always possible to modify a source string A into a target string B (provided that both strings are subject to the same alphabet).

$d_L[i, j] = \min (d_L[i,j-1]+c[\varepsilon,B_j], d_L[i-1,j]+c[A_i,\varepsilon], d_L[i-1,j-1]+c[A_i,B_j])$ for $i \geq 1$ and $j \geq 1$,

with $d_L[0, 0] = 0$ and $d_L[i,-1] = d_L[-1, j] = \infty$ where: A_i is the i^{th} element of string A, B_j is the j^{th} element of string B

Fig 2.1 Levenshtein Distance Calculation

The Levenshtein Distance of two strings A and B (denoted as d_L (A, B)) is computed on the minimum number of single character insertions, deletions and substitutions required to convert string A to string B. The d_L(A, B) can be easily calculated, which is given in Fig: 2.1. However, greater the Levenshtein distance, more mismatched strings are found. The algorithm provides a choice to select the value of the approximation that the tool accepts as desired similarity distance (e.g. if the user accepts zero as the desired approximation, then only one target words with the minimum edit distance will be returned, whereas if any user enters e.g. 2 as the desired approximation, then a set of target words is returned having a distance <= (minimum + 2) from the source word [53]. e.g. , close → closing, stir → stirred.

Second Approach of Lemmatization:

The second approach is the **Morphological Analyzer** which is based on "**finite state automata (FSA)**". A Finite State Transducer is a more generic term than a finite-state automaton (FSA). A FSA provides a formal language by defining a set of accepted strings, while a FST provides relations in-between sets of strings. A FST will read a set of strings on the input tape**,** generate a set of relations on the output tape and translate the contents of its input tape to its output tape, to generate another string on its output tape. [65, 66]

2.4 The Concept of Morphological Analysis

M.P. Schützenberger (1961) explained and implemented two sequential transducers (where the output of the first form is the input of the second) for generating a new word and for extracting morphemes [33]. **"Morphology"** concept had been first incorporated into the generative linguistics domain with **Chomsky (1970), Halle (1973), Aronoff (1976) and Siegel** (1979) [35], [34], [5]. Chomsky & Halle (1968) proposed ordered context-sensitive linguistic rules for describing the phonological component as a system of rules which, maps surface structures into phonetic representations [34], [35]. They formulated many phonological rules based on which, words are segmented into morphemes. The 7th rule of Chomsky described a derivational word's segmentation phonological rule which stresses a primary-stressed vowel in the context. e.g. $[CoVCo]_N/[erase]_V r]_N$ [5, 34,35]. According to linguistic experts' observations, highly irregular relationships exist in-between English derived nominals and their supposedly corresponding verbs (profess—professor—profession—professional; social—socialist—socialite) and but for some nouns, no verb exists in the lexicon (tuition but no *tut, motion but no *mote).

C. Douglas Johnson (1972) discovered that ordered phonological rules can be implemented with a cascade of FSTs (Finite State Transducer) if the rules are never applied to their own output [37]. Based on Harris's phonetic concept, (1974) model segmented lexical text into stems and affixes with minimal human intervention which is described in statistical stemming approach section.[37]

Kaplan & Kay (1980) rediscovered the findings of Johnson and Schützenberger. Ronald M. Kaplan and Martin Kay were putting rules into a more algebraic perspective than Johnson. According to Kaplan and Kay, formal languages were constructed with sets of strings and mathematical objects which are built from a finite alphabet E by the associative concatenation operation [38]. Formal language theory has classified string sets and the subsets of E*. Ordered n

tuples of strings: $X=<x_1,x_2\ldots x_n>$ & $Y=<y_1,y_2\ldots y_n>$ are shown here. Concatenation of X & Y: $X.Y=_{df} <x_1y_1,x_2y_2,\ldots x_ny_n>$ and this has the expected property that |X.Y| = |X| + |Y| , even if they have different length and $|X|=_{df} \sum_i | xi |$ [38].

Kimmo Koskenniemi (1984) invented 2-level-morphology [39]. They explained that morphology of a language is a set of rules which start from an underlying lexical representation, and transform it step by step until the surface representation is reached. Koskenniemi's two-level morphological process was the first practical general model in computational linguistics for the analysis of morphological complex languages.In continuation of Kimmo's model, **Lauri Karttunen and Kent Wittenburg (2003)** have developed a "Two-Level Morphological Analyzer" (first FST compiler) based on Kaplan's implementation of the finite-state calculus [38] [39]. This analyzer can handle low level English morphological variants such as plural in nouns and present continuous, past and past participle in a verb and some comparative (-er), superlative (-est), singular & plural genitive ('s) and verb-to-adjective marker (-able). But most of the derivative and nominalized morphed words are still unexplored.

GoldSmith (2001) developed Linguistica5 software for unsupervised learning of linguistic structure. Unsupervised learning revolves around morphological signature's objects in Linguistica. For example, {Ø, s} is a morphological signature set which is used in any sizable English datasets, with possible linked stems such as walk-, jump- (which entails that the words walk/walks, jump/jumps occur in the data) [41,42].

Krister Lindén (2008, 2009) attempted to model the transformation between base and inflected form part by part, but adopted a simpler, three-way split into prefix, stem and suffix. Linde´n mentioned that the model was implemented as a **cascade of Finite state** [47].

Statistical Morphological Analyzer:

Popular Statistical Stemming:

Haris' phonetic segmentation: Based on literature survey, it was studied that **Zellig. S. HARRIS** [32], University of Pennsylvania (1955) has first discussed about structural phonemes, utterance, and successor variety and word boundaries in the structural and linguistics domain. Z. S. Harris' process focused on breaking phonetic text into its constituent morphemes. Z. S. Harris' process had broken phonetic text into its constituent morphemes.

Margaret. A. Hafer and Stephen. F. Weiss's segmentation process uses certain statistical properties of a corpus (successor and predecessor letter variety counts) to indicate where words should be divided [15]. Consequently, this process is less reliant on human intervention than are other methods for automated stemming. The process is based on the notion of a letter successor variety.

Successor Variety Stemming [15]: This stemmer is processed by decomposing word into segment and is developed based on structural morphology to identify the word and morpheme boundaries through phonemes' distribution in a large body of utterances [15]. The word is segmented into "stems and affixes" by using certain statistical properties of a corpus (successor and predecessor variety counts) which decide the index from where the word should be parsed. Hafer's and Weiss's three approaches to generate stem [15] from a word are discussed here.

I. Cut off method**:** -The simplest method to decompose a test word, is first to choose some cut off "k" and then decompose the word at a particular index where its successor (or predecessor or both) variety meets or over reaches k. This method was adopted by Haris in his phonetic analysis. In this method, careful decision to select 'k' is required [15].

"Test Word: READABLE

Corpus: ABLE, APE, BEATABLE, FIXABLE, READ, READABLE, READING, READS, RED, ROPE, RIPE. [15]

Threshold=2 is considered and segmentation will be done when successor variety >= threshold" [11]. In "Readable" word, "Read" segment has 3 successor varieties (A, I, S) and next all segments (reada, readab, readabl, readable) have only one successor variety. So, from "readable, "read" segment is generated which will become a stem token.

II. Complete Word Method: A word will be decomposed at a particular index before that, if that word-prefix or suffix is observed to be a whole word in the corpus.If a shortest substring "w'' of a word "w' " exists as a word in a document, is considered a stem of that particular word (w) [15].

"READ" is a broken part (segment) from "READABLE" word which is identified as a complete word in the corpus [15].

III. Entropy method: It exploits of Lemmatizations the benefits of the dissemination of successor variety characters.

The method's working process is as follows: Let $|D\alpha_i|$ be the number of words in a text body beginning with the i length sequence of letters of a test word α

Let $|D\alpha_{ij}|$ be the number of words in $|D\alpha_i|$ has the successor j.

Then probability that the successor letter of α_i is the j^{th} letter of the alphabet is given by $\frac{|D\alpha ij|}{|D\alpha i|}$

The entropy of successor system for a test word prefix α_i is $H\alpha_i = \sum_{\rho=1}^{26} -\frac{|D\alpha ij|}{|D\alpha i|} . \log_2 \frac{|D\alpha ij|}{|D\alpha i|}$

The importance of each successor letter is measured by entropy calculation within the segmentation process. Entropy is calculated by successor letter's probability of occurrence. Rare successor letters will have a smaller impact on taking decisions in segmentation with compare to highly probable successor variety. [15]

Determination of stems: After segmentation of a word, which segment will be considered as a stem, will be determined. In most cases the first segment is chosen as the stem. When the segment appears in many different words, it is probably a prefix. Then, the second segment should be considered as a stem.[15]

Hafer and Weiss used the following rule: if (first segment occurs in <= 12 words in corpus)

First segment is considered as stem else (the second segment will become a stem) [15]

N-gram Segmentation's overview is already given in the "Introduction" chapter**.** It is applied for classification of text. The number of shared digram within a pair of character strings is used to compute Dice's Similarity Coefficient which helps to develop cluster from sets of character strings within text [14]. A digram is a pair of successive letters. There is a little bit of perplexity in using the "stemming" term here because this method is not at all generating stem [14]. Similarity-measure of two word forms is calculated based on the number of shared unique digram of a word with another word's unique digram which are generated in this method. Association measures between the pair of terms "phosphorus" & "phosphate" are calculated here.

Phosphorus=>ph, ho, os, sp, ph, ho, or, ru, us ; Unique digrams = ph, ho, os, sp, or, ru, us ;

Phosphate=>ph, ho, os, sp, ph ha, at, te ; Unique digrams = ph, ho, os, sp, ha, at, te ;

Dice's coefficient (similarity) $S = \frac{2C}{A+B} = \frac{2*4}{7+7} = .57$ [14]

A=number of unique digrams in the 1st word B = number of unique digrams in the 2nd word

C = number of unique digrams shared by A, B

Table 2.7 Similarity Matrix Based on Dice's similarity

Words	Statistics	Statistic	statistical	statistician
Statistics	0.00	1.08	0.93	0.87
Statistic	0.00	0.00	1.00	0.93
Statistical	0.00	0.00	0.00	0.81
Statistician	0.00	0.00	0.00	0.00

The above mentioned similarity sparse matrix justifies the observation of Adamson and Boreham. Cluster is generated based on cut-off value 0.6 in the similarity matrix. So, this process helps to cluster the documents based on similarity measure in between adjacent words by creating virtual digram.

Statistical YASS Stemming: Prasenjit Majumder, Mandar Mitra, Swapan K. Parui, and Govinda Kole, ISI (2007) have proposed a **stemming algorithm-YASS** [12], independent of linguistic expertise. This stemming aims to **identify morphological classes that share common roots**. In this article, researchers look at the problem of stemming for such resource-poor languages (in particular, for Bengali). Purely unsupervised statistical clustering techniques which do not assume any language-specific information have been proposed for this purpose. This **Stemming [YASS]** has a target to identify morphological classes that share common roots based on some statistical string similarity measures which is implemented in our proposed algorithm [12]. In this YASS stemming algorithm, average of summation of four String Distance Measures functions are used to cluster words into homogeneous groups. Each group is expected to represent an equivalence class consisting of morphological variants of a single root word.

Third Approach of Lemmatization:

This approach is called "**Ripple Down Rule**". It is a data structure, used by popular Lemmagen [53], which allows retrieving a possible lemma of a given inflected or derivational form. Ripple-down rules are an incremental approach to knowledge acquisition. Ripple-down rules are an incremental approach to knowledge acquisition.

RDR ::= IF rule.condition THEN rule.class EXCEPT rule.exceptions

Where – rule.condition is a wordform suffix, – rule.class is a transformation to be applied to a wordform, and – rule.exceptions ::= nil | RDR+ (list of RDRs) [55].

Fourth Approach of Lemmatization:

The fourth approach is **Affix lemmatizer** which is a combination of a rule based and supervised training approach and the last approach is fixed-length truncation approach. German and Dutch need more advanced methods than suffix replacement since their affixing of words

(inflection of words) can include both prefixing, infixing and suffixing. Therefore, a trainable lemmatizer is required to create that handles pre- and infixes in addition to suffixes [54]. Following Dutch full form lemma pair: Afgevraagd→afvragen. (Translation: wondered, to wonder). Input given to the training program, it should produce a transformation rule like this:*ge*a*d → ***en. [54]

Fifth Approach of Lemmatization:

The fifth one is **Fixed Length Truncation** approach which has been successfully executed for the Turkish language. In this approach, word is simply truncated and the first 5 and 7 characters of each word are considered as its lemma. In this approach, words with less than n characters are used as a lemma with no truncation. This fixed prefix truncation technique gives good results in the Turkish language. Can et.al investigated the fixed-length truncation for 3 through 7 characters; they found that 5 were the best result. According to other experiments carried out on approximately 24K Turkish words, the length of Turkish words are composed of an average of 7.07 letters. So this technique uses the first 5 and 7 characters. [55]

Sixth Approach of Lemmatization:

The sixth one is **Context Sensitive lemmatization technique. Abhisek Chakrabarty Onkar Arun Pandit and Utpal Garain**, ISI (2017) developed **"Context Sensitive lemmatization model using Two Successive Bidirectional Gated Recurrent Networks**" which is based on supervised deep neural network architecture and language- independent context sensitive lemmatization technique. This model learns the transformation patterns between word-lemma pairs (sang->sing, achieving->achieve) and hence, can handle the unknown word forms too [49].

Toms Bergmanis and Sharon Goldwater (2018) developed **Context Sensitive Neural Lemmatization with Lematus** which was executed on inflected English word variants to determine their dictionary root word (e.g., swims, swimming, swam, swum) and also on data from 20 typologically varied languages. They presented Lematus, a simple sequence-to-sequence neural machine translation framework that uses character-level context [50].

Abhisek Chakrabarty, Akshay Chaturvedi and Utpal Garain, CVPR Unit, ISI (2019) developed language independent **CNN-based Context Sensitive Lemmatization** which was executed on two indic languages (Hindi, Marathi) and two European languages (French and Spanish). For the English language, experiment was only done for verb in different tenses [51].

Seventh Approach of Lemmatization:

The seventh one is a Dictionary-based Turkish Lemmatizer (DTL) [58]. DTL is based on "Grand Turkish Dictionary" as a dictionary. For the sake of efficiency, DTL uses radix-trie, which is one of the most efficient variants of the trie data structure. **One dictionary-based approach is WordNet Lemmatizer**. [66]

2.5 Conclusion and Summary

Features and shortcoming of all the above-mentioned stemming, lemmatizers and morphological analyzers were studied and observed after implementation and execution of available algorithms. Observation assists to design the proposed models "LemmaChase" and "LemmaQuest". In the next chapter, newly developed algorithm for comparative studies of existing popular stemmers and comparative output are discussed.

Chapter 3
Comparative Study of Stemmers and Lemmatizer

3. Comparative Study of Stemmers and Lemmatizers

The algorithm is developed for analyzing the performance of popular stemmers with respect to degree of correctness of the generation of the stem and execution time required to generate output. Comparative output after execution of existing lemmatization applications is shown in this chapter and studies of all those applications are also discussed here.

3.1 Algorithm for Comparative Study of Stemmers

The algorithm is designed and implemented for generating comparative output of Porter, Lovins and Paice stemming techniques [64]. The steps of the algorithm are given below:

1. Punctuation, numeric values, stop words and proper nouns are removed after tagging by Stanford Part-of-Speech software tool and then filtered collection of words are used as an input of this comparative algorithm. For any text, duplicate words are also initially removed. DUC text (Document Understanding Conferences) [72] and dictionary words are taken for this comparative study.
2. Filtered sorted unique words are accepted for executing this algorithm [64]. Input word is processed by the program in which Lovins, Porter, Paice and KStem techniques are implemented.

Features of Lovins Stemming: Most of the irregular singular-plural morphed words are correctly stemmed here. Based on the execution of this algorithm on some DUC .doc files, average computational time (in milliseconds) to generate stem token is lowest with compare to other affix removal stemmers.

Features of Porter Stemming: Size of the vocabulary is reduced by one third with the help of this suffix stripping stemmer.

Features of Paice Stemming: Based on generation of stem token with the help of this comparative algorithm, calculated ICF (Index Compression factor) of Paice is quite high with

compare to other affix removal stemmers which is explained in below table. So, based on ICF value**,** Paice is stronger stemmer with compare of others.

Feature of KStemming: It is observed that KStem generates root- dictionary words instead of generating meaningless stem token for morphed input word in many cases.

"N = Number of unique words before stemming, S = Number of unique stems after stemming, ICF = Index Compression factor; Calculation of ICF = (N-S)/N; Greater the value of ICF, greater will be strength of the stemmer."[20]

3.2 Comparative Output of Stemmers

Below mentioned empirical result analyzes the degree of strength of popular stemmers with respect to correctness in generation of stem and in computation time.

Table 3.1 Comparative Result of Affix-Removal Stemmers

Text [72]	Actual Size	Effective Size	ICF	Time	ICF	Time	ICF	Time	ICF	Time
AP880911-0016 [DUC]	322	103	Porter		Lovins		Paice		Krovetz (KStem)	
			0.029	21.6	0.067	11.6	0.048	10.6	0.038	95
AP880912-0095 [DUC]	782	256	0.066	23.2	0.07	19.8	0.085	13	0.046	100.4
AP880912-0137 [DUC]	666	217	0.059	26.2	0.073	12.8	0.078	13.4	0.0414	100.2

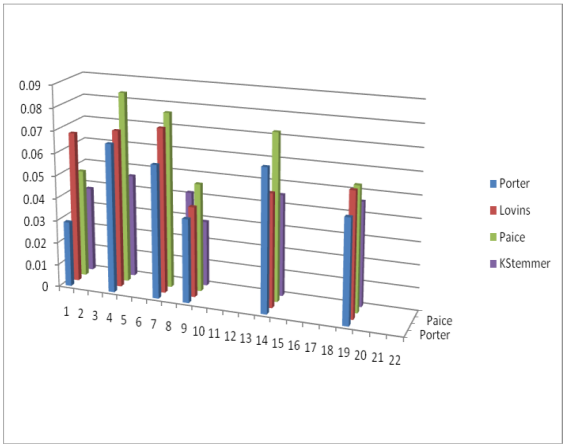

Fig: 3.1 3-D Bar Diagram for depicting value of Stemmers.

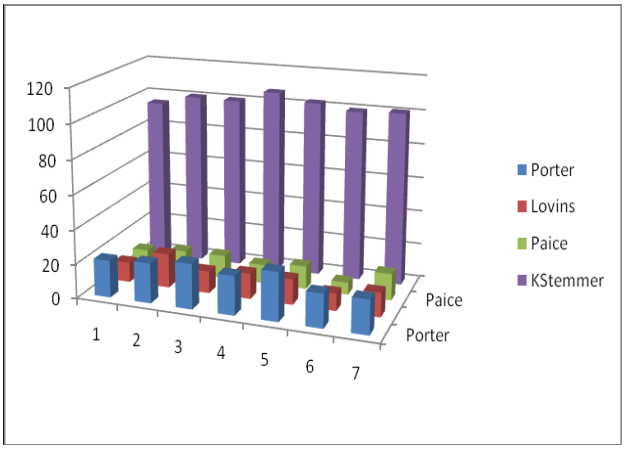

Fig: 3.2 3-D Bar Diagram for depicting ICF & Processing Time of Stemmers

Table 3.2 Number of Stem Word Generation for a set of Wordlist

List of Morphed Words	Number of Stemmed terms				List of Morphed Words	Number of Stemmed terms			
	Porter	Lovins	Paice	KStemmer		Porter	Lovins	Paice	KStemmer
Abilities Ability	1	1	1	2	Abolish, Abolished Abolishes Abolishing Abolition	2	3	2	2
Abnormal Abnormally	1	1	1	1					

3.3 Popular Lemmatization Tools

1) MorphAdorner: It is a command-line Java application which executes as a pipeline manager for processes performing morphological adornment of words in a text. It lemmatizes inflected words into their corresponding lemma [59].

2) **LemmaGen:** It was trained to generate accurate and efficient lemmatisers for twelve different languages. Its analysis on the corresponding lexicons depicts that LemmaGen outperforms the lemmatisers generated by two alternative approaches, RDR and CST, both in terms of accuracy and efficiency [53]. But Lemmagen only lemmatizes inflected words.

3) **Spacy Lemmatizer:** It comes as a pre-built model of NLP tool that can parse text and compute various NLP-related features through one single function call [69].

4) **UDPipe:** It is a trainable pipeline, executing for tagging, lemmatization and dependency parsing of CoNLL-U files. UDPipe is a language-agnostic tool [70].

5) **Stanford Lemmatizer:** Stanford morphological analyzer (LemmaProcessor) [2005] are developed [61]. Based on KIMMO & GOLDSMITH's approaches which are already discussed in the previous chapter. Stanford developed CoreNLP for natural language processing in Java. CoreNLP facilitate users to acquire linguistic annotations for text, including token and sentence boundaries, parts of speech, named entities, lemmatization, dependency and constituency parses, coreference, sentiment, quote attributions, and relations. The centerpiece of CoreNLP is the pipelines which take raw text as an input. [61]. Below Fig3.3 depicts the flow of CoreNLP.

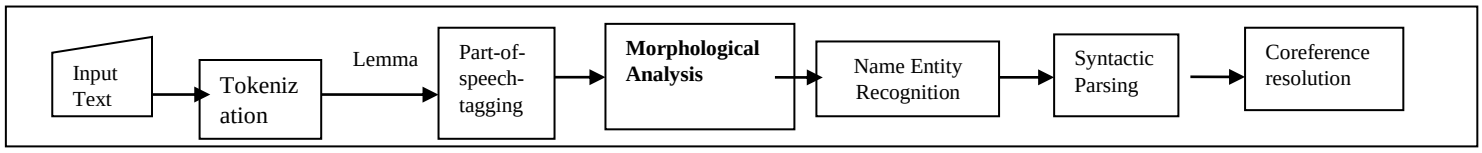

Fig. 3.3 Stanford CoreNLP Model

6) **BioLemmatizer**: It (2012) is also a lemmatization tool which handles bio-medical terms as well as some derivational adjectives and adverbs, based on a morphological analyzer, MorphAdorner 2.0. [73]

Other popular available online Lemmatizers are Wordnet Lemmatizer with NLTK and CST's online tools.

3.4 Comparative Output of Existing Lemmatizers

After execution of all below-mentioned lemmatization applications, it is noticed that all applications are able to generate the correct lemma for all inflected words, but do not attempt to generate lemma for derived and nominalized words. BioLemmatizer is able to handle biological terms perfectly.

Table 3.3 Output for Nominalized words by existing Lemmatizers

<table>
<tr>
<td>1. LemmaGen Lemmatizer :
http://lemmatise.ijs.si/Services
Input: “Acceptance achievement Action activity achievement judgment addition adjustment admiration amazement annoyance attention”
Output: “Acceptance achievement Action activity achievement judgment addition adjustment admiration amazement annoyance” [No Change]</td>
<td>2. BioLemmatizer
Input : Output
anlagen, NN : anlage
spermatogonia, NN : spermatogonium</td>
</tr>
<tr>
<td>3.Stanford CoreNLP XML Output
Sentences
Sentence #1 Tokens
<table>
<tr><th>Id</th><th>Word</th><th>Lemma</th><th>Char begin</th><th>Char end</th><th>POS</th></tr>
<tr><td>1</td><td>Acceptance</td><td>acceptance</td><td>0</td><td>10</td><td>NN</td></tr>
<tr><td>2</td><td>Achievement</td><td>achievement</td><td>12</td><td>23</td><td>NN</td></tr>
<tr><td>3</td><td>Action</td><td>action</td><td>25</td><td>31</td><td>NN</td></tr>
<tr><td>4</td><td>Activity</td><td>activity</td><td>33</td><td>41</td><td>NN</td></tr>
<tr><td>5</td><td>Activeness</td><td>activeness</td><td>43</td><td>53</td><td>NN</td></tr>
</table></td>
<td>4
English Lemmatizer Example
Enter a spelling and optionally a word class in the input boxe spellings before lemmatization.
Spelling: application
Standardize spelling
Primary word class: noun
Secondary word class:
Lexicon:
Early Modern English
Eighteen Century English
Nineteenth Century Fiction
Lemmatize Clear
Lemmatizer Results
Lemma: application
Lancaster stem: apply
Porter stem: applic
MorphAdorner</td>
</tr>
</table>

3.5 Conclusion and Summary

In this chapter, output of popular affix-removal stemmers and lemmatizers were generated and observed the correctness of all those stemmers and lemmatizers for a list of words . The error list was also observed. Based on limitations of all these stemmers and lemmatizers, the proposed model LemmaChase is decided to develop to overcome all those limitations.

Chapter 4
LemmaChase: Lemmatizer

4. Overview of LemmaChase Model

The proposed model LemmaChase has been specially developed to generate the correct lemma for nominalized words and to convert a text into limited numbers of lemmas. This chapter is describing the steps of LemmaChase and depicting LemmaChase's output which is also compared with existing lemmatizers' output.

4.1 Steps of LemmaChase

The broad view of LemmaChase is depicted in below-mentioned flowchart.[63]

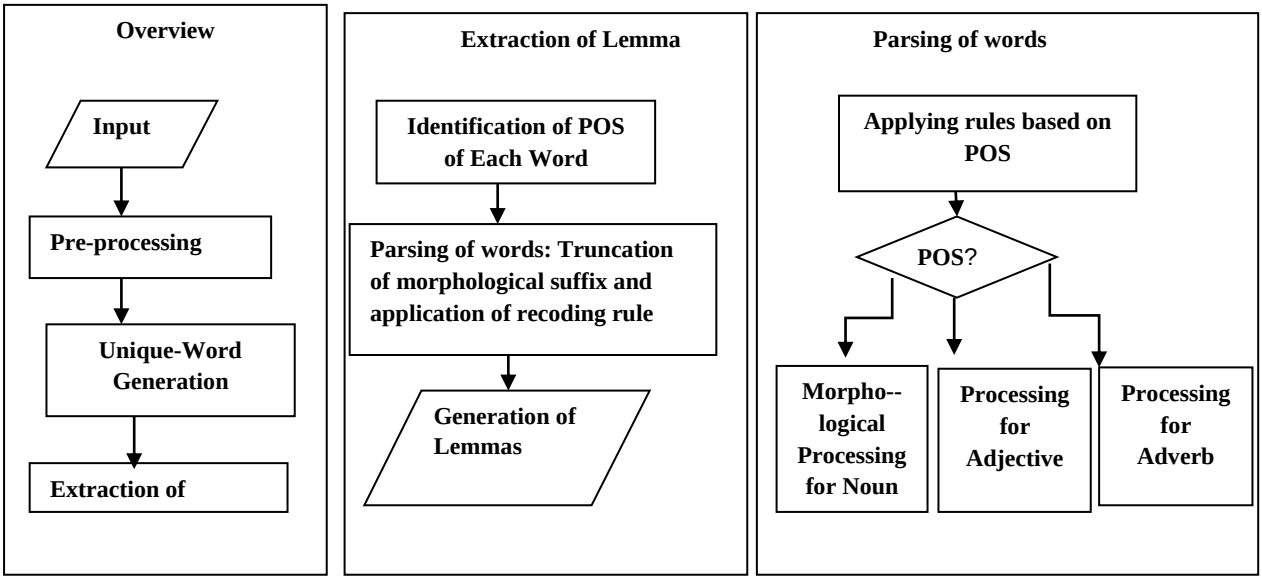

Fig. 4.1 Flowchart of LemmaChase

Preprocessing of the File:

- Remove all unnecessary characters. In this step, all unnecessary characters like punctuations, symbols will be removed. Convert all words into lower case. Remove all stop words.
- Tokenize the sentence into words.

- Generate sorted unique word list.
- Tag all unique words using Stanford POS tagger.

4.2 Algorithm of LemmaChase

In this proposed lemmatization model, following steps are designed to get lemmas from their surface words. [63]

Based on token's POS, words are parsed with the help of morphological rules to get their corresponding lemma.

1. Proper Noun tagged word is not processed further, same input word will be identified as its lemma. e.g., word "India" has identified as a lemma for input word "India". But, "Indian" is identified as a Noun and an Adjective both that will be parsed further.
2. If a word's POS is identified only as Verb in any tense using JWNL API and WordNet dictionary, extracted lemma for above mentioned input-word is finalized as final lemma. e.g., words "achieved", "went" and "programmed" exist as only verb POS in the dictionary. Lemma "achieve", "go" and "program" are respectively finalized as the lemma for the given sequence of input-words.
3. If the input word is identified as both Noun and Verb POS form, lemma of Verb form's word is selected as lemma if and only if length of lemma <= length of input word. e.g., "programming" and "left" exist as both VERB and NOUN.
4. If the input word always exists as a noun in any dictionary and its extracted lemma is the same as that word itself, then such a condition indicates that this lemma could be correct or incorrect. For nominalized noun, the correct lemma cannot be extracted using WordNet dictionary. The following steps should be implemented in such cases. Lovins' stemming [9] "Longest-match" principle is applied on input-word for suffix/endings identification (e.g. -al, -al after –I, -al with duplicate characters, -ance, -ence, -ion, -ication,-ion, -ism, -ship, -(p)tion, -ure, -ment, -age, -cation, -ief) and recoding rules are applied for restructuring input words.

   i) "The longest-match principle states that within any given class of endings, if more than one ending provides a match the one which has the longest match, should be removed." In the case of suffix "-ion" and "-ation", suffix "-ation" will get more priority to be removed [9].

ii) If trimmed stem-token exists as VERB POS word, its lemma is extracted and selected as lemma of input-word. If the trimmed stem-token does not exist as a valid dictionary word in any POS form, then new token is generated using the recoding rule.

The recoding rules are different for different class of endings for generating a new token which is listed in below-mentioned table.

If the token exists as a valid dictionary word and also exists either as a VERB or as a NOUN form, the lemma of the word which is in VERB form is selected as the final lemma of the input-word.

If it is only in NOUN form (e.g., history, world), then the input word is selected as the final lemma. If the token does not exist as a valid dictionary-word, original input word is selected as a lemma of input word.

5. If input word is identified as both Adjective and Verb POS form, lemma of Verb form's word is selected as the proposed model's lemma. If input word is only identified as Adjective (e.g., nice, happy and historical) and its extracted lemma is identical with the input-word, such a condition indicates that this lemma is either correct or incorrect. For nominalized adjectives, correct lemma cannot be extracted using WordNet dictionary. Lovins' stemming [9] "Longest-match" principle is again applied for adjectives (e.g. -able, -en, -ious, -ful, -AL, -ary, -ic, -ical, -y, -ed) and recoding rules are also applied for adjectives.

6. If the input word only exists as an Adverb POS in any dictionary (e.g., gracefully, gently) and it may exist as a nominalized adverb. After applying ADVERB suffix (e.g. -ly, -ally,-ably, -ically, -wise, -y) truncation rules and recoding rules, if a new dictionary word is created, its' lemma is selected for input adverb words.

   If a new dictionary word is not created, the word (e.g., instead) is selected as lemma and input word is not a nominalized word.

With the help of above mentioned steps and using the suffix and recoding rules mentioned in below-mentioned tables, maximum allied nominalized words as well as allied morphed words are merged into their corresponding dictionary base words or morphological root words which is called lemma.

With the help of this proposed model, generation of new dictionary root-word is also possible from any morphological derived input word. Normal (bird-birds/box-boxes) and irregular

(fungus-fungi/woman-women) singular-plural, verb in any tense (run-ran/go-went) and nominalized words (acceptance-acceptability/application-applicability/ add-addition) can merge to their root words/lemmas using this proposed model .

Table 4.1 Verb Related Suffix Rules [63]

	Suffix	**Input word**	**Lexical function within the item-and-process model**
Verb To Noun	-ation	don-ation, regul-ation, educ-ation	**[x]$_V$->[[x]$_V$*ation*]$_N$:**[don<u>ate</u>]$_V$ation]$_N$,[regul<u>ate</u>]$_V$ation]$_N$, educ<u>a</u>te]$_V$ion]$_N$
	-al	approv-al, arriv-al, revers-al, refus-al	**[x]$_V$->[[x]$_V$*al*]$_N$: A**prov<u>e</u>]$_V$al]$_N$, [arriv<u>e</u>]$_V$al]$_N$, [reverse]$_V$al]$_N$
	-er	teach-er, runn-er, writ-er, build-er, paint-er	**[x]$_V$->[[x]$_V$*er*]$_N$,**:[[teach]$_V$*er*]$_N$,[[run]$_V$*er*]$_N$, [build]$_V$*er*]$_N$
	-ist	cycl-ist, typ-ist, copy-ist	**[x]$_V$->[[x]$_V$*ist*]$_N$,**:[[cycle]$_V$ist]$_N$,[[typ<u>e</u>]$_V$ist]$_N$, [cop<u>y</u>]$_V$ist]$_N$
	-ment	pave-ment, appoint-ment, govern-ment,	**[x]$_V$->[[x]$_V$*ment*]$_N$:** [appoint]$_V$*ment*]$_N$,
	-ee	employ-ee, detain-ee, pay-ee, intern-ee	**[x]$_V$->[[x]$_V$*ee*]$_N$:** [employ]$_V$ee]$_N$
Verb To Adj	-ise/-ize	real-ise, neutral-ise, fertil-ise, immun-ise	**[x]$_V$->[[x]$_V$*ise*]$_{ADJ}$:** [[neutral]$_V$ise]$_{ADJ}$,[[fertil<u>e</u>]$_V$ise]$_{ADJ}$,
	-ive	act-ive, evas-ive, product-ive,	**[x]$_V$->[[x]$_V$*ive*]$_{ADJ}$:** [[act]$_V$ive]$_{ADJ}$, [[product]$_V$ive]$_{ADJ}$
	-able	read-able, govern-able; manage-able	**[x]$_V$->[[x]$_V$*able*]$_{ADJ}$:** [read]$_V$able]$_{ADJ}$, [govern]$_V$able]$_{ADJ}$
Noun To Verb	-ate	regul-ate, capacit-ate, don-ate	**[x]$_N$->[[x]$_N$*ate*]$_V$:**[[regul<u>ar</u>]$_N$ate]$_V$, [[capacity]$_N$ate]$_V$
	-ise/ -ize	colon-ise, American-ise, computer-ise	**[x]$_N$->[[x]$_N$*ise*]$_V$:**[[computer]$_N$ise]$_v$
	-ify	carbonify , historify, personify, solidify	**[x]$_N$->[[x]$_N$*ify*]$_V$:** [[carbon]$_N$ify]$_V$, [[history]$_N$ify]$_V$,
Verb To Verb	-er/-fy	chatt-er, patt-er, flutt-er, sign-ify	**[x]$_V$->[[x]$_V$*er*]$_V$:** [[chat]$_V$er]$_V$, [[flut]$_V$er]$_V$, [[sign]$_V$ify]$_V$

Table 4.2 Adjective Related Suffix Rules [63]

	Suffix	**Input word**	**Lexical function within the item-and-process model**
Adj To Adj	-ish	narrow-ish, blu-ish, pink-ish	**[x]$_{ADJ}$->[[x]$_{ADJ}$*ish*]$_{ADJ}$:**[[narrow]$_{ADJ}$ish]$_{ADJ}$, [[blue]$_{ADJ}$ish]$_{ADJ}$
Noun To Adj	-al	division-al, medicin-al, origin-al, univers-al	**[x]$_N$->[[x]$_N$*al*]$_{ADJ}$:** [[medicin<u>e</u>]$_N$al]$_{ADJ}$, [[origin]$_N$al]$_{ADJ}$
	-ate	intim-ate, accur-ate, obdur-ate,	**[x]$_N$->[[x]$_N$*ate*]$_{ADJ}$:**[[intim]$_N$ate]$_{ADJ}$, [[accur]$_N$ate]$_{ADJ}$

4.3 Tools used in the LemmaChase model

POS-tagger [61], JWNL tool [81] and WordNet dictionary [66] are used to identify the context and to extract the lemma of a word. These tools assist LemmaChase model to generate the correct lemma for any input word.

Part-of-Speech Tagger: Stanford Lemmatizer uses Penn Treebank for tagging Part-of-Speech of each input-word of proposed model. Stanford POStagger is used in the proposed model "LemmaChase" and "LemmaQuest" lemmatizer for identifying POS of each word.

Table 4.3 Sample Input Text for POS Tagging

"Eight Dead, Up to 18 Missing After Explosion at Marine Band ComplexEds: LEADS with 7 grafs to UPDATE with eight dead, Scotland Yard sending anti-terrorist unit to investigate; pickup 7th graf pvs, `Ten doctors..."

Table 4.4 Output of Stanford POS-Tagger

"Eight/CD Dead/JJ ,/, Up/IN to/TO 18/CD Missing/VBG After/IN Explosion/NN at/IN Marine/NNP Band/NNP ComplexEds/NNPS :/: LEADS/VBZ with/IN 7/CD grafs/NNS to/TO UPDATE/VB with/IN eight/CD dead/JJ ,/, Scotland/NNP Yard/NNP sending/VBG anti-terrorist/JJ unit/NN to/TO investigate/VB ;/: pickup/NN 7th/JJ graf/NN pvs/NNS ,/, `/`` Ten/CD doctors/NNS .../: '/"

Penn Treebank: The Penn Treebank, in its eight years of operation (1989-1996), produced approximately 7 million words of part-of-speech tagged text, 3 million words of skeletally parsed text, over 2 million words of text parsed for predicate argument structure, and 1.6 million words of transcribed spoken text annotated for speech disfluencies. All available Penn Treebank materials are distributed by the Linguistic Data Consortium. [80]

Table 4.5 Sample of POS Tag Set [80]

POS tag	POS tag	POS tag
1. CC Coordinating conjunction	10.LS List item marker	19.PP Possessive pronoun
2. CD Cardinal number	11.MD Modal	20.RB Adverb
3. DT Determiner	12.NN Noun, singular or mass	21.RBR Adverb, comparative
4. EX Existential there	13.NNS Noun, plural	22.RBS Adverb, superlative
5. FW Foreign word	14.NNP Proper noun, singular	23.RP Particle
6. IN Preposition/subord.	15.NNPS Proper noun, plural	24.VBG Verb, gerund/present participle
7. JJ Adjective	16.PDT Predeterminer	25.VBP Verb, non-3rd ps. sing. present
8. JJR Adjective, comparative	17.POS Possessive ending	26.VBD Verb, past tense
9. JJS Adjective, superlative	18.PRP Personal pronoun	27.VB Verb, base form

JWNL Tool: JWNL is an API for accessing WordNet-style relational dictionaries. It also provides functionality for relationship discovery and morphological processing. First, JWNL.initialize() is invoked to initialize the code of a program. Then, Dictionary.getInstance() is called to get the currently installed dictionary. [81]

WordNet: WordNet was developed by Princeton University. WordNet 2.1 is used in this research work. "WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet superficially resembles a thesaurus, in that it groups words together based on their meanings. Structure: Synonyms words that denote the same concept and are interchangeable in many

contexts are grouped into unordered sets (synsets). Each of WordNet's 117000 synsets is linked to other synsets by means of a small number of "conceptual relations." [65]

4.4 Result and Discussion

Above mentioned Stanford POS-tagger is executed on the input-text file for attaching POS-tag with each filtered input word. After generating a new morph from any inflected or derived input word with the help of LemmaChase, morph will be verified with the help of JWNL executable jar file which assists to proceed further for constructing a new lemma.

Output of LemmaChase:

Table 4.6 Output of LemmaChase

Sequence	Word Noun	Output of LemmaChase ((12% error)	Word Adjective	Output of LemmaChase(11% error)	Word Adverb	Output of LemmaChase
1.	acceptance	Accept	acceptable	Accept	angrily	Angry
2.	achievement	Achieve	achievable	Achieve	academically	Academy
3.	action(Verb)	Act	Acting	Act	rightly	Right
4.	activity	Act	actionable	Act	busily	Busy
5.	legalization	Legal	academic	academy	legally	Legal
6.	judgment	Judge	judgmental	Judge	hopefully	Hopeful
7.	addition	Add	additional	Add	easily	Easy
8.	adjustment	Adjust	adjustable	Adjust	idly	Idle
9.	admiration	Admire	admirable	Admire	artistically	Artistic
10.	amazement	Amaze	amazing (Verb)	Amaze	rapidly	Rapid
11.	annoyance	Annoy	annoying (Verb, Noun)	Annoy	tragically	Tragic
12.	attention	Attend	attentive	Attend	classically	Classic
13.	attraction	Attract	attractive	Attract	really	Real
14.	advisement	Advise	advisable	Advise	luckily	Lucky
15.	avoidance	Avoid	avoidable	Avoid	magically	Magical
16.	comfort(Verb)	Comfort	comfortable	Comfort	politically	Politics
17.	cheerfulness	Cheer	cheerful	Cheer	maturely	Mature
18.	confusion	Confuse	confusable	Confuse	voluntarily	Volunteer

Lemma generated by LemmaChase is shown in below-mentioned table. For derived and nominalized Noun, Adjective and Adverb, output is generated with 11-12% error which is nominal. [63]

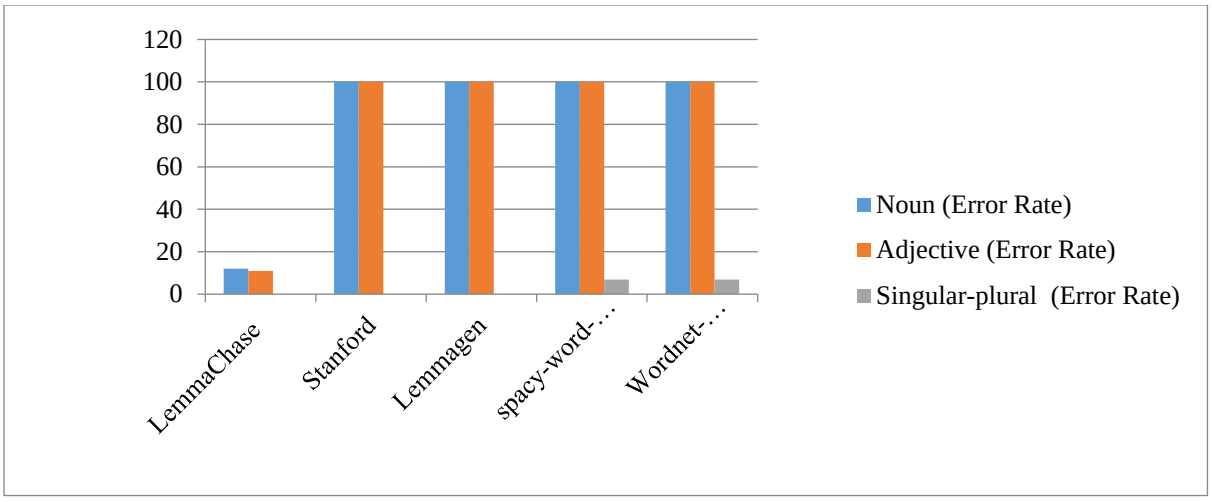

Fig 4.2 Comparison of Rate of Error in Generation Lemma

4.6 Conclusion and Summary

It is observed that the proposed lemmatizer-LemmaChase handles maximum number of nominalized words which are not still handled by available popular lemmatizers. It handles most of the morphed and derived words (singular, plural in Noun, irregular singular-plural/ verbs /adjective/ adverbs) with any POS form in a text. In this proposed model, single class morphed or derived words are merged into a single lemma, but this model processes same class morphed words separately to get a single lemma repeatedly, which leads to consuming more processor time. When words “application”, “applications”, “applicant”, “applicants”, “applicable” and “applicability” are present in a text, all these six allied words are parsed and processed individually to generate the lemma “apply” six times. So, this work is extended by statistical computation to create an individual group for a single class of morphed words which can easily reduce the processing task of identification of lemma. Placing correct class morphed words into the proper group leads avoiding error in lemma generation. With the help of the next proposed model LemmaQuest, number of word processing steps will be drastically decreased to generate lemma. It can automatically minimize the overhead of WordNet dictionary look-up.

Chapter 5
LemmaQuest: Lemmatizer

5. Overview of LemmaQuest Model

LemmaQuest Model has attempted to cover up the inefficiency of LemmaChase which was observed in execution. LemmaQuest is an effective model for all those texts in which the maximum number of allied morphed words co-exist (e.g. dictionary words). This chapter first shows a diagram for depicting the major steps of LemmaQuest.

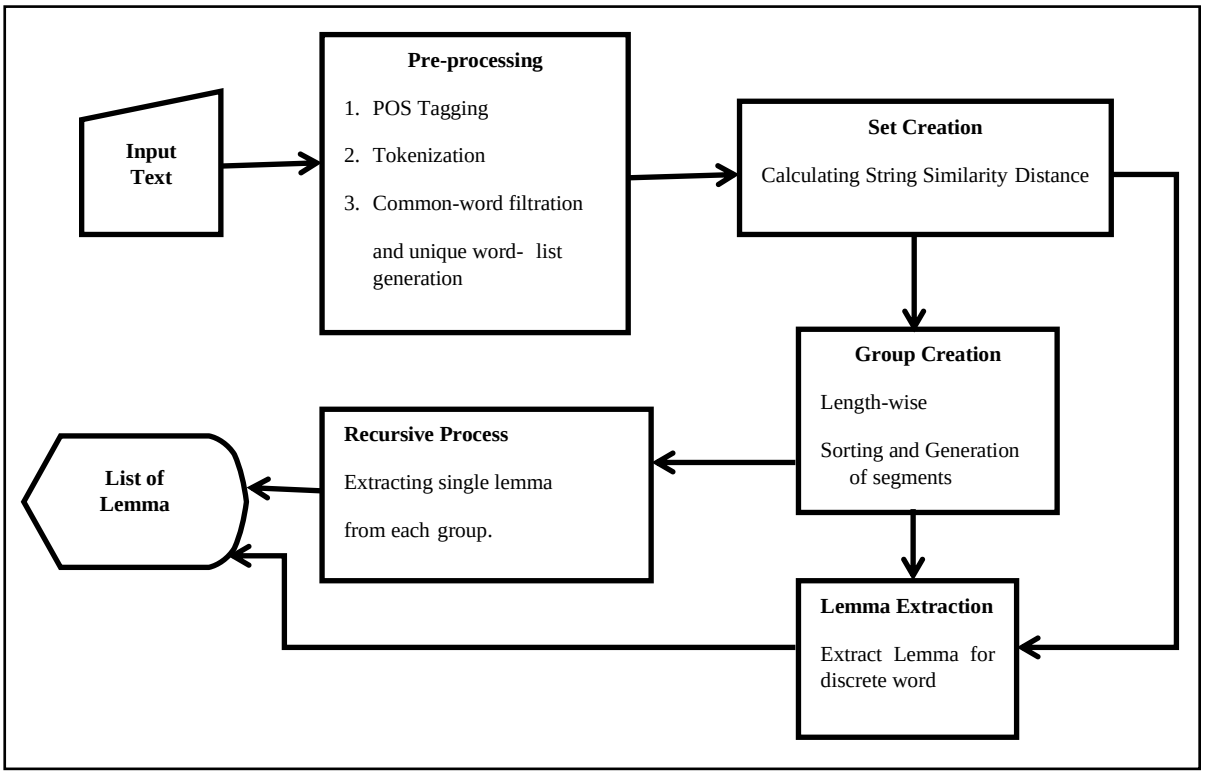

Fig 5.1 LemmaQuest Model

5.1 Algorithm of LemmaQuest

LemmaQuest model is depicted in Fig. 1. In the LemmaQuest, w_s is considered as an input word to be lemmatized. w_s is processed to get lemma through various operations, which are described below.

Prerequisites' tasks for LemmaQuest are required the followings:

1. List [w_s, POS_tag] indicates a list of sorted filtered unique words with their contextual POS (POS_tag).

2. $f_{POS}(w\_s)$ returns the part-of-speech (POS) of the word "w_s" based on current context, using Stanford POS-tagger. LemmaQuest uses three resources:

- Stanford-Part-of-speech-Tagger: maximumEntropyPOSTagger(List[w_s]) returns List of words (w_s) with their contextual POS [$f_{POS}(w\_s)$].
- WordNetDictionary: Method "morphology. lookupBaseForm (POS.VERB/ NOUN/ADJ/ ADV, w_s)" returns IndexWord of w_s. "IndexWord.getLemma()" returns base_word/ lemma of input word w_s [32].

Let D = {d_1, d_2, . . . ,d_n} be the set of all dictionary words present in the WordNet dictionary.

A. **Steps of LemmaQuest:**

Input: A surface word w_s, List [w_s, POS_tag] & POS(w_s),List={w_i/w_i_POS_tag, w_j/ w_j_POS_tag,w_n/ w_n_POS_tag : w_s/w_s_ POS_tag }; here {w_i, w_j….w_n} ≡ of w_s .

Steps:

Step: 1. Build two set S_p and S_p' from a list of inputs.

i) Build a set **S_p** (S_p ⊂ D) which is a collection of [w_i, w_j….w_n] and POS(w_i) , POS(w_j) where Sim(w_i, w_j) < δ (δ is a string similarity threshold value) and POS(w_s) ∈ $POS_{dic}(d_i)$, ∃ w_i ∈ w_s in WordNet-dictionary.

S_p ={ {w_i, POS(w_i), w_j, POS(w_j), Sim(w_i, w_j)}, {w_i, POS(w_i), w_{j+1}, POS(w_{j+1}), Sim(w_i, w_{j+1})},……}.

Word-pair-similarity Distance [Sim(w_i, w_j)] : Sim() is a method which is derived from "YASS measure". where i=1 to (n-1): j=2 to n (n indicates number of w_s are available in the List) Compute method: Sim(w_i, w_j);

In LemmaQuest, "Distance measure" is calculated for each character mismatch of w_i and w_j instead of only 1st character mismatch (implemented in YASS).

$$D1(X,Y) = \sum_{i=0}^{n}\left(\frac{1}{2^i}\right) p_i \quad (1)$$

$$D2(X,Y) = \frac{(1)}{m} \times \sum_{i=m, xi\#yi}^{n} \frac{1}{(2^{i-m})} \quad \text{if } m>0,\alpha \text{ otherwise} \quad (2)$$

$$D3(X,Y) = \frac{(n-m+1)}{m} \times \sum_{i=m, xi\#yi}^{n} \frac{1}{(2^{i-m})} \quad \text{if } m>0,\alpha \text{ otherwise} \quad (3)$$

$$D4(X,Y) = \frac{(n-m+1)}{n+1} \times \sum_{i=m, xi\#yi}^{n} \frac{1}{(2^{i-m})} \quad \text{if } m>0,\alpha \text{ otherwise} \quad (4)$$

$$(\text{Average\_DIST}(X,Y)) = \frac{(D1 + D2 + D3 + D4)}{4} \quad (5)$$

$$(\text{Sim}(w_i, w_j)) \equiv (\text{Average\_DIST}(X,Y))$$

ii) Build a set S_p' ($S_p' \subset D$). S_p' is a set which has a collection of distinct words where Sim (w_i, w_j) = ∞ (infinite), no word is bounded with any other S_p'={ w_i, w_j, ...}

Step: 2. Build possible number of groups of allied morphed words, derived from S_p set.

G= {G_1, G_2, G_m}:

where m is a number of groups which are formed from S_p. $G_i \subset S_p$ where 1>= i <=m

1st word w_i is considered as a key word in S_p set and then identify all those w_j words associated with key word w_i, where Sim(w_i, w_j) < δ.

For distinct w_i, its' all morph-allied w_j are piled up in a single group G_i.

Recursively, for next $w'_{(i+1)}$, all allied $w'_{(j+1)}$ s' are piled up in a single group G_{i+1}.

Here, $\forall$ w_s , m number of group ($G_1 \ldots G_m$) are created where m<n and 1<=i<=m

G_i = { {w_i, POS(w_i), flag}, {{w_{i+1}, POS(w_{i+1}), flag},……}

Step: 3. First, sort all words of group G_i based on word length and then sort alphabetically in such a way, shortest length word will be popped up on the top of each group. The group G_i will be refined as G_i' after rearrangement of words.

w_s_t = $(stem)_p$ + s_x, morphed word with the shortest suffix will be popped on top of G_i', where w_s_t is the top most word of G_i'. G_i' = { w_s_t, w_i | $w_i \in w\_s$}

Step: 4. Segment all allied morphed words (w_s) of G_i into $stem_p$ after applying Hafer's & Weiss's "Word Segmentation By Letter Successor Varieties" method.

Pseudo Code of segmentation of Step-4 is given below:

```
main()
{
for w_s in the group Gi'
{Segment the word into 2 parts using minimum length substring;
array=
SegmentUsingSuccessor
(left_semented_ part); }}
```

```
SegmentUsingSuccessor(stem)
{
s=stem;
for each substring s of each w_s {
Calculate the successor count Sn;
if found a local peak/plateau
Save this position to an array of split points;
 return array}
```

Here "Peak and plateau" and "complete word" techniques are applied for segmentation of w_s.

Step: 5. If segmented $stem_p$ is found as a complete dictionary word, then $stem_p$'s lemma is extracted; otherwise lemma of w_s_t is extracted.

Step: 6. If extracted Lemma (obtained from step 5) is found character-to-character identical with its' input word ($stem_p$/ w_s_t) or it has only one character dissimilarity with its' input word only in the last index-position, then this condition indicates that extracted lemma may not be the correct

lemma for the input-word. e.g., incorrect lemma "application" /"development" /"employee" (L_1) is extracted for input-word "applications"/"developments"/"employee" (w_s) using WordNet. e.g."$[employee]_{Noun}$->$[employ]_{Verb}$" is targeted to identify by LemmaQuest.

i) LemmaQuest parses derived-word or nominalized word (w_s_t/$stem_p$) into stem based on below mentioned POS-class-wise suffix rules. Then, stem ($stem_p$**)** will be recoded to generate new words using below-mentioned suffix-based recoding rules.[-ational->-ate], [-ment→-]. Reconstruction of stem into lemma is already discussed into "LemmaChase" lemmatization process in the previous chapter. Truncation of the longest suffix and the addition of the context-sensitive new suffix will be incorporated for the generation of a new word from existing input surface-word. e.g. rel~~ational →~~rel<u>ate</u> [63].

A new word's lemma is extracted and it (L_i) represents as a final Lemma of individual group G_i'. $L_i \equiv G_i'$ where G_i'= {w_s}: w_s indicates morphed words.

ii) Otherwise, lemma (L_i) getting from step 5, is concluded as the final lemma of G_i'

Step: 7. Within set S_p', all morphologically disconnected words are clustered based on the value of Sim (w_i, w_j). So LSV technique is not feasible to apply to all those words. LemmaQuest parses the words and applies POS-based derivative suffix rules and recoding rules to extract root-word/lemma for a particular inflected or derived input surface-word w_s of S_p' [63].

Step: 8. Here, ∀ w_s _(input-words), set 'S_L' is developed such that S_L ={ L_1,L_2,L_3.....}. Output: S_L represents a lemma list for all input-words (List[w_s]).

Observed Output: $S_L = f_{LemmaQuest}$(List[w_s])

After parsing of input-words and then reconstruction of the stem into a new valid dictionary word, LemmaQuest is able to extract the correct lemma from the maximum number of English derivational surface words and nominalized words.

5.2 Result of LemmaQuest

LemmaQuest generates list of lemmas for a set of input words. It generates a single lemma for a group of allied morphed words and also generates individual lemma for an individual word within a set, where morphologically dissimilar words exist.

A. **Input**: List of sample input of allied morphed surface-words (w_s) which is extracted from dictionary is shown in Table 5.1. Note: Stanford POS-tagger is executed for ∀ w_s.

Table 5.1 Sample Input Text

Morphed Word Collection
Abruptly abruptness/ Absconded absconders absconding absconds /Absence absences absented absentee absentees absenting absently absents absentia / Absorbed absorbency absorbent absorber absorbers absorbing absorption/ Academia academic academically academics academies/ Accented accenting accents accentual accentuate/ Acceptable acceptably acceptance acceptor acceptors/ Accessed accesses accessible accessibly accessing accessory/ Accidence accidental accidents/ Acclaimed acclaimers acclaiming acclaims /Achievable achieved achiever achievers achieves achieving/ Acidic acidities acidity acidly acidosis acidified acidifier acidifiers acidifies acidify acidifying/ Algebra algebraic algebras/ Algeria Algerian Algerians

Output: Groups and Lemmas (SL={L1, L2, L3....}) are given below mentioned table.

Note: JWNL Tool is executed for validating the morph which is created by LemmaQuest.

Table 5.2 Output-Groups and their Lemmas

Group of allied morphed words (G_i')	Lemma (L_i)
abruptly RB , abruptness JJ	Abrupt
Absconds JJ, absconded VBD, absconders NNS, absconding VBG	Abscond
absence NN absents NNS absences NNS absented VBD absentee NN absentia JJ absently RB absentees NNS absenting VBG	Absent
absorbed VBN absorber NN absorbent JJ absorbers NNS absorbing VBG absorbency NN	Absorb
academia NN academic JJ academies NNS academically JJ	Academy
accents NNS accented VBD acceptor NN accessed VBD accenting VBG accentual JJ accentuate NN	Accent, accept
Acceptor NN acceptor NN acceptable JJ acceptable RB acceptance NN	Accept
Accessed VBD accesses NNS accessing VBD accessory NNS accessible JJ accessibly RB	Access
accidence NN accidents NNS accidental JJ	Accident
acclaims NNS acclaimed JJ acclaimers NNS acclaiming JJ	Acclaim
Aced VBD acesVBZ	Ace
Acidic JJ acidly RB acidify NN acidity NN acidified VBD acidifier NN acidifies VB acidifiers NNS acidifying VBG	Acid
Algebra NN Algeria FW algebras FW Algeria JJ algebraic JJ algerians NNS	Algebra, Algeria

B. **Input**: A DUC text (w_s) [72] is shown in Table 5.3. Note: Stanford POS-tagger is executed for ∀ w_s.

1. Assign POS tag with each word of the text.

Table 5.3 Input-DUC Text

DUC Text Sample-Word Collection with POS
Gilbert/NNP Reaches/VBZ Jamaican/JJ Capital/NNP With/IN 110/CD Mph/NNP Winds/NNP Hurricane/NNP Gilbert/NNP ,/, packing/NN 110/CD mph/NN winds/NNS and/CC torrential/JJ rain/NN ,/, moved/VBD over/IN this/DT capital/NN city/NN today/NN after/IN skirting/VBG Puerto/NNP Rico/NNP ,/, Haiti/NNP and/CC the/DT Dominican/NNP Republic/NNP ./. There/EX were/VBD no/DT immediate/JJ reports/NNS of/IN casualties/NNS ./. Telephone/NNP communications/NNS were/VBD affected/VBN ./. ``/`` Right/RB now/RB it/PRP 's/VBZ actually/RB moving/VBG over/IN Jamaica/NNP ,/, "/" said/VBD Bob/NNP Sheets/NNP ,/, director/NN of/IN the/DT National/NNP Hurricane/NNP Center/NNP in/IN Miami/NNP ./. ``/`` We/PRP 've/VBP already/RB had/VBN reports/NNS of/IN 110/CD mph/NN winds/NNS on/IN the/DT eastern/JJ tip/NN ./. ``/`` It/PRP looks/VBZ like/IN the/DT eye/NN is/VBZ going/VBG to/TO move/VB lengthwise/NN across/IN that/DT island/NN ,/, and/CC they/PRP 're/VBP going/VBG to/TO bear/VB the/DT full/JJ brunt/NN of/IN this/DT powerful/JJ hurricane/NN ,/, "/" Sheets/NNPS said/VBD ./. Forecasters/NNS say/VBP Gilbert/NNP was/VBD expected/VBN to/TO lash/VB Jamaica/NNP throughout/IN the/DT day/NN and/CC was/VBD on/IN track/NN to/TO later/RB strike/VB the/DT Cayman/NNP Islands/NNPS ,/, Forecasters/NNS at/IN the/DT center/NN said/VBD the/DT eye/NN of/IN Gilbert/NNP was/VBD 140/CD miles/NNS southeast/NN of/IN Kingston/NNP at/IN dawn/NN

today/NN ./. Maximum/NNP sustained/VBD winds/NNS were/VBD near/IN 110/CD mph/NN ,/, with/IN tropical-storm/NN force/NN winds/NNS extending/VBG up/RP to/TO 250/CD miles/NNS to/TO the/DT north/NN and/CC 100/CD miles/NNS to/TO the/DT south/NN ./. Sunday/NNP night/NN :/: '/" Cuba/NNP 's/POS official/JJ Prensa/NNP Latina/NNP news/NN agency/NN said/VBD a/DT state/NN of/IN alert/NN was/VBD declared/VBN at/IN midday/NN in/IN the/DT Cuban/JJ provinces/NNS of/IN Guantanamo/NNP ,/, Holguin/NNP ,/, Santiago/NNP de/IN Cuba/NNP and/CC Granma/NNP ./. In/IN the/DT report/NN from/IN Havana/NNP received/VBD in/IN Mexico/NNP City/NNP ,/, Prensa/NNP Latina/NNP said/VBD civil/JJ defense/NN officials/NNS were/VBD broadcasting/VBG bulletins/NNS on/IN national/JJ radio/NN and/CC television/NN recommending/VBG emergency/NN measures/NNS and/CC providing/VBG information/NN on/IN the/DT storm/NN ./. Flights/NNS were/VBD canceled/VBN Sunday/NNP in/IN the/DT Dominican/NNP Republic/NNP ,/, where/WRB civil/JJ defense/NN director/NN Eugenio/NNP Cabral/NNP reported/VBD some/DT flooding/NN in/IN parts/NNS of/IN the/DT capital/NN of/IN Santo/NNP Domingo/NNP and/CC power/NN outages/NNS there/RB and/CC in/IN other/JJ southern/JJ areas/NNS ./.

2. Allied morphed words' groups are developed dynamically to extract single root word.

Table 5.4 Output-Groups and their Lemmas

Group of allied morphed words (G_i')	**Lemma (L_i)**
Agency , agencies	agency
Alert, alerted	Alert
Appeals, appears, appear	Appeal, appear
Broadcast, broadcasting	broadcast
Cause, causing	cause
Center, central	center
Coast, coastal	coast
Danger, dangerous	danger
East, eastern	east
Flood, flooding, flooded	flood
Forecaster, forecasters	forecaster
Head, heads, heavy	Head, heavy
High, higher	high
Hurricane, hurricanes	Hurricane
Jamaican, Jamaicans	Jamaica
Move, moved, moves	move
North, northwest, northward	north
Official, officiate	office
Pack, packed, packing, parish	Pack, parish

Table 5.5 Output-Collection of Lemmas of discrete words (Set: S_p')

Root	**word**		**Root**	**word**
actual	actually		Casualty	casualties
add	adding		Communicate	communications
affect	affected		Defend	defense
arrive	arrived		Discontinue	discontinued
begin	began		Govern	government
board	boarding		Inform	information
bind	bound		Instruct	instructions
branch	branches		Meteorology	meteorologist
brush	brushing		News	news
bulletin	bulletins		Prepare	preparation
cancel	canceled		Vacation	vacationer

5.3 Comparative Study

Results of LemmaQuest are compared with output of existing popular Lemmatizers and morphological analyzer. This comparative study depicts that the number of correct lemma generations from LemmaQuest is quite low compared with others. Fig 5.2 depicts the number of

lemma extracted by LemmaQuest and by other existing lemmatizers from a set of allied morphed dictionary words. It has been observed that optimized and the best number of lemmas are generated by LemmaQuest. e.g. LemmaQuest extracts 80 lemmas, whereas 215, 200 and 299 numbers of lemmas are extracted by Stanford tool, Spacy and WordNet Lemmatizer respectively out of 364 allied-morphed dictionary based input-word list. Fig 5.3 depicts the percentage of errors that occurred in the extraction of lemma from input-word by all available lemmatizers including LemmaQuest. All normal verbs in different tenses, regular-irregular singular-plural and any other inflected words are mostly lemmatized perfectly by below mentioned lemmatizers. But, all available lemmatizers are not able to parse derived words and nominalized words (derived noun and verb), adjective and adverb to extract their corresponding lemma, but such type of lemmatization is mostly done by LemmaQuest. In case of lemmaQuest, 12% error is generated, whereas the rest of the lemmatizers generates almost 100% error in extraction of lemma for all derived surface-words. MorphAdorner can parse some adverb POS words accurately just like lemmaQuest.

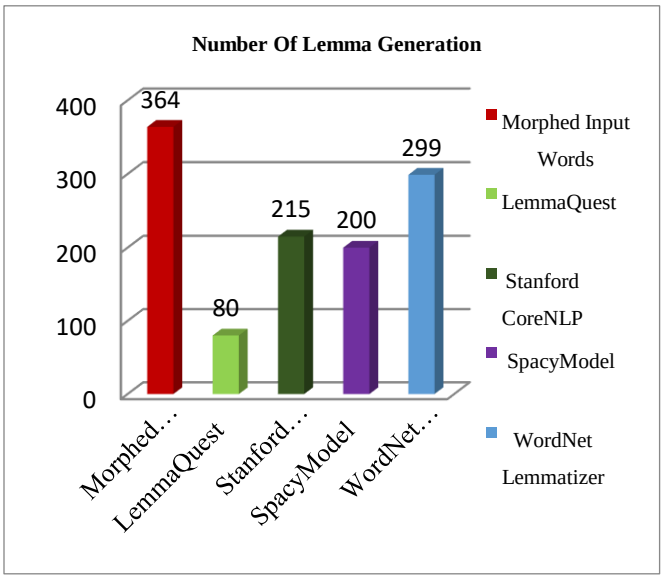

Fig. 5.2 Lemma Generation

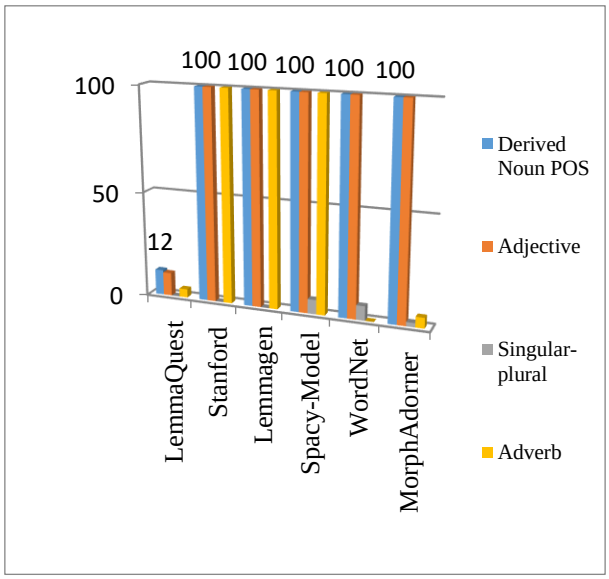

Fig. 5.3 Comparison of Rate of Error in Generation Lemma

Table 5.6 Comparative Output of Different Lemmatizers with LemmaQuest

No	Noun, Verb, Adjective, Adverb	Lemma Quest	Stanford Lemmatizer	WordNet Lemmatizer	Spacy-word-lemmatizer	Lemmagen	MorphAdorner Morphological Analyzer	CTS's Lemmatizer
		Lemmas						
1	abilities	ability	Ability	ability	Ability	ability	ability	Ability
2	formulae	formula	formula	formula	Formulae	formula	formula	Formula
3	insurance	insure	insurance	insurance	Insurance	insurance	insurance	Insurance
4	sang	sing	Sing	sing	Sing	sing	sing	Sing
5	women	woman	woman	woman	Woman	woman	woman	Woman
6	existence	exist	existence	existence	Existence	existence	existence	Existence
7	education	educate	education	education	Education	education	education	Education
8	employment	employ	employment	employment	Employment	employment	employment	Employment
9	government	govern	government	government	Government	government	government	Government
10	baggage	bag	baggage	baggage	Baggage	baggage	baggage	Baggage
11	legislature	legislate	legislature	legislature	Legislature	legislature	legislature	Legislature
12	angrily	angry	angrily	angrily	Angrily	angrily	angry	Angrily
13	academically	academy	academically	academically	Academically	academically	academically	Academically
14	academics	academy	academics	academics	Academics	academics	academics	Academics
15	employee	employ	employee	employee	Employee	employee	employee	Employee
16	magically	magic	magically	magically	Magically	magically	magical	Magically
17	artistically	artist	artistically	artistically	Artistically	artistically	artistical	Artistically
18	signify	sign	signify	signify	Signify	signify	signify	Signify
19	universally	universe	universally	universally	Universally	universally	universal	Universally
20	comfortably	comfort	comfortably	comfortably	Comfortably	comfortably	comfortable	Comfortably

5.4 Conclusion and Summary

LemmaQuest handles maximum number of English derived morphed words and nominalized words, along with inflected words to extract their corresponding lemma. When a text file or dictionary morphed word list is accepted as an input, lemma extraction, processing is not required to apply on individual input-words in this model. For a group of all allied morphed words, single run for extraction of a lemma is required. One single lemma would be representative for all allied morphed words of a group in this model. Corpus-based stemmer helps to prevent to conflate "policy / police", "addition/ additive", "university"/universal" all these un-related words into a single stem token. In the proposed LemmaQuest model, all these above-mentioned semantically un-related words are clustered in a single group which will lead to miss a few words to extract their lemma. Such type of error has also been handled by LemmaQuest with pre-requisite knowledge of POS of a word. This is effective when a text contains a large number of allied derived words. This model has specially focused on nominalized words to find out their correct lemma which exists in a particular POS, different from nominalized word's POS. LemmaQuest also extracts lemma for non-allied morphed words

in individual word-level. This lemma extraction technique is effectively useful in information extraction. e.g. text simplification, text-summarization and keyword extraction.

5.5 Future Work

This work can be extended for Indian regional languages with incorporation of language dependent suffixes and recoding rules with aid of regional-WordNet.

5.6 Reference

1. CHAPTER 8: STEMMING ALGORITHMS: W. B. Frakes Software Engineering Guild, Sterling, VA 22170
2. Frakes William B. “Strength and similarity of affix removal stemming algorithms”. ACM SIGIR Forum, Volume 37, No. 1. 2003, 26-30.
3. DAWSON, J. 1974. "Suffix Removal and Word Conflation." ALLC Bulletin, Michelmas, 33-46.
4. https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html
5. Mark Aronoff and Kirsten Fudeman , What is Morphology? ISBN: 978-1-405-19467- 9, Wiley-Blackwell, October 2010.
6. Carmen Galvez, Fe´lix de Moya-Anego´n and Vı´ctor H. Solana. In stemming and lemmatization processes, two term conflation approaches: non-linguistic and linguistic : Department of Information Science, University of Granada, Granada, Spain.Journal of Documentation.Vol. 61 No. 4, 2005. pp. 520-547
7. https://en.wikipedia.org/wiki/Christopher_D._Paice
8. Anjali G. Jivani. A Comparative Study of Stemming Algorithms**.** IJCTA | NOV-DEC 2011. ISSN:2229-6093
9. J. B. Lovins. Development of a stemming algorithm,” Mechanical Translation and Computer Linguistic., vol.11, no.1/2, pp. 22-31. 1968.
10. https://en.wikipedia.org/wiki/Julie_Beth_Lovins
11. Prasenjit Majumder, Mandar Mitra, Kalyankumar Datta: Statistical vs. Rule-Based Stemming for Monolingual French Retrieval, 2006 Adhoc Monolingual Information Retrieval
12. Prasenjit Majumder, Mandar Mitra, Swapan K. Parui, and Govinda Kole, YASS: Yet another suffix stripper. Article in ACM Transactions on Information Systems, 2007
13. Michela Bacchin *, Nicola Ferro, Massimo Melucci *: A probabilistic model for stemmer generation: Department of Information Engineering, University of Padova, Italy, Information Processing and Management 41 (2005) 121–137 123
14. Adamson, G.W., Boreham, J.: The use of an association measure based on character structure to identify semantically related pairs of words and document titles
15. M. Hafer And Stephen F. Weiss. Word Segmentation by Letter Successor Varieties:Pergamon Press. Printed in Great Britain: Vol. 10, pp. 371-385, 1974.

16. Benno Stein and Martin Potthast : Successor Variety for Stemming" in 2007.
17. Porter M.F. "An algorithm for suffix stripping". Program. 1980; 14, 130-137.
18. Porter M.F. "Snowball: A language for stemming algorithms". 2001.
19. Paice Chris D. "Another stemmer". ACM SIGIR Forum, Volume 24, No. 3. 1990, 56-61.
20. Paice Chris D. "An evaluation method for stemming algorithms". Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval. 1994, 42-50.
21. Xu Jinxi and Croft Bruce W. "Corpus-based stemming using co-occurrence of word variants". ACM Transactions on Information Systems. Volume 16, Issue 1. 1998, 61-81.
22. Funchun Peng, Nawaaz Ahmed, Xin Li and Yumao Lu. "Context sensitive stemming for web search". Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. 2007, 639-646.
23. Jiaul H. Paik, Swapan K. Parui (ISI, Kolkata) :A Novel Corpus-Based Stemming Algorithm using Co-occurrence Statistics: Publication: SIGIR '11: Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval July 2011 Pages : 863–87
24. Krovetz Robert. "Viewing morphology as an inference process". Proceedings of the 16th annual international ACM SIGIR conference on Research
25. Hull D. A. and Grefenstette,. "A detailed analysis of English Stemming Algorithms", XEROX Technical Report, http://www.xrce.xerox.
26. Juan-Manuel Torres-Moreno: Beyond Stemming and Lemmatization: Ultra-stemming to Improve Automatic Text Summarization, 2012:Laboratoire Informatique d'Avignon, BP 91228 84911, Avignon, Cedex 09, France
27. R. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval. Addison Wesley, 1999.
28. C.D. Manning and H. Schütze. Foundations of Statistical Natural Language Processing. The MIT, Press, 1999.
29. Harman Donna. "How effective is suffixing?" Journal of the American Society for Information Science. 1991; 42, 7-15 7.
30. https://en.wikipedia.org/wiki/Morphological_derivation
31. https://devopedia.org/stemming
32. Harris Z.S. Distributional Structure. In: Papers in Structural and Transformational Linguistics. Formal Linguistics Series. Springer, 1955. ISBN978-94-017-6059-1
33. M.P. Schützenberger. 1961. A remark on finite transducers. Information and Control, 4:185–187. 1961
34. Morris Halle. Prolegomena to a Theory of Word Formation. Linguistic Inquiry Volume 4 Number i (Winter, I973) 3-I6.

35. NOAM CHOMSKY, MORRIS HALLE. The Sound Pattern of English. Harper & Row, Publishers, New York, Evanston, and London, 1968.
36. Francis Katamba. English Words: by Routledge: ISBN 0-415-10468-8 (pbk) , 1994.
37. C. Douglas Johnson. Formal Aspects of Phonological Description. Mouton. 1972
38. Ronald M. Kaplan* & Martin Kay. Regular Models of Phonological Rule Systems. Xerox Palo Alto Research Center and Stanford University, 1980
39. Kimmo Koskenniemi. Two-level morphology: A general computational model for word-form recognition and production: Publications/Department of General Linguistics, University of Helsinki, ISBN-10: 9514532015, 1984
40. Eric Gaussier.Unsupervised learning of derivational morphology from inflectional lexicons. Xerox research Center Europe 6, chemin de Maupertuis 38240 Meylan F, 1999.
41. John Goldsmith. Unsupervised Learning of the Morphology of a Natural Language:StudyReport: Computational Linguistic: ISSUE 2, pp153-198, 2001.
42. John Goldsmith Linguistica 5: Unsupervised Learning of Linguistic Structure, 2001
43. LAURI KARTTUNEN AND KENNETH R. BEESLEY. Twenty-five years of finite-state morphology: CSLI Publications, pp.71-83. July 13, 2005
44. Sylvain Neuvel. Whole Word Morphologizer: Expanding the WordBased Lexicon: University of Chicago. A Nonstochastic Computational. Brain and Language 81, 454–463. 2002
45. Richard Wicentowski**.** Modeling and Learning Multilingual Inflectional Morphology in a Minimally Supervised Framework. Baltimore, Maryland, 2002
46. Douglas W. Oard, Gina-Anne Levow, and Clara. Cabezas. CLEF Experiments at Maryland: Statistical Stemming and Backoff Translation. Print ISBN: 978-3-540-42446-8, 2001.
47. Krister Lindén A Probabilistic Model for Guessing Base Forms of New Words by Analogy. Springer, Berlin, Heidelberg. Online ISBN: 978-3-540-78135-6.
48. Jiaul H. Paik, Mandar Mitra, and Swapan K. Parui. GRAS: An Effective and Efficient Stemming Algorithm for Information Retrieval. ACM Transactions on Information Systems, Vol. 29, No. 4, Article 19. 2011
49. Abhisek Chakrabarty Onkar Arun Pandit and Utpal Garain. Context Sensitive Lemmatization Using Two Successive Bidirectional Gated Recurrent Networks. Conference: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1)
50. Toms Bergmanis and Sharon Goldwater (2018). Context Sensitive Neural Lemmatization with Lematus. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) P: 1391–1400
51. Abhisek Chakrabarty, Akshay Chaturvedi and Utpal Garain CNN-based Context Sensitive Lemmatization. Publication Proceedings of the ACM India Joint International Conference on Data Science and Management of Data. Pages 334–337, 2019

52. Dimitrios P. Lyras, Kyriakos N. Sgarbas, Nikolaos D. Fakotakis(2007). Using the Levenshtein Edit Distance for Automatic Lemmatization: A Case Study for Modern Greek and English: 19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007), Vol-2. Pp.428-435.
53. Matjaž Juršič, Igor Mozetič, Tomaž Erjavec, Nada Lavrač. LemmaGen: Multilingual Lemmatisation with Induced Ripple-Down Rules: Journal of Universal Computer Science, vol. 16, no. 9, 1190-1214, 2010.
54. Bart Jongejan , Hercules Dalianis: Automatic training of lemmatization rules that handle morphological Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 145–153, Suntec, Singapore, 2-7 August 2009.
55. Information Retrieval on Turkish Texts , FAZLI CAN, , SEYIT KOCBERBER, ERMAN BALCIK , CIHAN KAYNAK, H. CAGDAS OCALAN, ONUR M. VURSAVAS :'Information retrieval on Turkish texts.' Journal of the American Society for
56. D.R. Morrison, "PATRICIA-Practical Algorithm To Retrieve Information Coded in Alphanumeric," J. ACM, vol. 15, no. 4, pp. 514
57. 'Information retrieval on Turkish texts.' Journal of the American Society for Information Science and Technology. Vol. 59, No. 3 (February 2008), pp. 407-421."
58. Richard Wicentowski**.** Modeling and Learning Multilingual Inflectional Morphology in a Minimally Supervised Framework. Baltimore, Maryland, 2002
59. http://morphadorner.northwestern.edu/morphadorner/
60. https://cst.dk/tools/index.php?lang=en
61. Christopher D. Manning: Linguistics & Computer Science: Stanford University, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, David McClosky. The Stanford CoreNLP Natural Language Processing Toolkit
62. Rupam Gupta, Anjali G. Jivani. Analyzing the Stemming Paradigm: Information and Communication Technology for Intelligent Systems (ICTIS 2017) - Volume 2 pp 333-342. 2017
63. Rupam Gupta, and Anjali G. Jivani. LemmaChase*:* A Lemmatizer: International Journal on Emerging Technologies 11(2): 817-824. 2020
64. Dr. Anjali Ganesh Jivani , Rupam Gupta. Empirical Analysis of Affix Removal Stemmers: et al, Int.J.Computer Technology &Applications, Vol 5 (2),393-399 IJCTA. 2014
65. https://wordnet.princeton.edu/
66. Jorge Morato, Miguel Ángel Marzal, Juan Lloréns, and José Moreiro . WordNet Applications: Petr Sojka, Karel Pala, Pavel Smrž, Christiane Fellbaum, Piek Vossen (Eds.): GWC 2004, Proceedings, pp. 270–278. 2004.

67. Michela Bacchin *, Nicola Ferro, Massimo Melucci *: Department of Information Engineering, University of Padova, via Gradenigo, 6/a, Padova 35131, Italy A probabilistic model for stemmer generation , Information Processing and Management 41 (2005) 121–137
68. Chapin, Paul: Syntax of word-derivation in English
69. http://textanalysisonline.com/spacy-word-lemmatize
70. Milan Straka and Jana Strakova´: Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe, Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics, CoNLL 2017
71. WORDS & TRANSDUCERS: Speech and Language Processing: An introduction to speech recognition, natural language processing, and computational linguistics. Daniel Jurafsky & James H. Martin. Copyright c 2007, All rights reserved. Draft of October 12, 2007.
72. DUC text. http://www-nlpir.nist.gov/projects/duc/data.html
73. Haibin Liu, Tom Christiansen. (2012). BioLemmatizer: a lemmatization tool for morphological processing of biomedical text. Published in J. Biomedical Semantics. Computer Science, Medicine Journal of Biomedical Semantics, DOI: 10.1186/2041-1480-3-3. Corpus ID: 15618342.
74. https://en.wikipedia.org/wiki/Finite-state_transducer
75. http://ufal.mff.cuni.cz/udpipe
76. https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
77. https://en.wikipedia.org/wiki/Information_extraction
78. https://en.wikipedia.org/wiki/Information_retrieval
79. https://en.wikipedia.org/wiki/Natural_language_processing
80. https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
81. Computing with a Thesaurus Word Senses and Word Relations: https://web.stanford.edu/~jurafsky/slp3/slides/Chapter18_introandsimilarity.pdf

5.7 Publication

1. Rupam Gupta , Dr. Anjali Ganesh Jivani ,. Empirical Analysis of Affix Removal Stemmers: et al, Int.J.Computer Technology &Applications, Vol 5 (2),393-399 IJCTA. 2014
2. Rupam Gupta, Anjali G. Jivani. Analyzing the Stemming Paradigm: Information and Communication Technology for Intelligent Systems (ICTIS 2017) - Volume 2 pp 333-342. 2017
3. Rupam Gupta, and Anjali G. Jivani. LemmaChase*:* A Lemmatizer: International Journal on Emerging Technologies 11(2): 817-824. 2020

- Rupam Gupta, Anjali G. Jivani : LemmaQuest Lemmatizer: A Morphological Analyzer Handling Nominalization (Submitted in two journals & two conferences)