TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

THESIS NO: 072-MSCS-660

A RULE BASED STEMMER FOR NEPALI

BY

PRAVESH KOIRALA

FINAL REPORT

SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SYSTEMS AND KNOWLEDGE ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

NOVEMBER, 2017

A

THESIS

ON

A RULE BASED STEMMER FOR NEPALI

BY:

PRAVESH KOIRALA

072/MSCS/660

SUPERVISED BY:

Dr. AMAN SHAKYA

A THESIS SUBMITTED TO DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SYSTEM AND KNOWLEDGE ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

INSTITUTE OF ENGINEERING, PULCHOWK CAMPUS

TRIBHUVAN UNIVERSITY

LALITPUR, NEPAL

NOVEMBER, 2017

Copyright ©

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certify that it has been read and recommended to the Department of Electronics and Computer Engineering for acceptance, a report of thesis entitled **"A Rule Based Stemmer For Nepali"**, submitted by **Mr. Pravesh Koirala** in partial fulfillment of the requirement for the award of the degree of **"Master of Science in Computer System and Knowledge Engineering"**.

_____

Supervisor, Dr. Aman Shakya
Lecturer
Department of Electronics and
Computer Engineering,
Central Campus,
Institute of Engineering.

_____

External Examiner, Dr. Manish Pokharel
Associate Professor
Department of Computer Science
and Engineering
Kathmandu University

_____

Committee Chairperson, Prof. Dr. Subarna Shakya
Professor
Department of Electronics and Computer Engineering,
Central Campus,
Institute of Engineering.

Date of Approval:

Departmental Acceptance

The thesis entitled **"A Rule Based Stemmer For Nepali"**, submitted by **Mr. Pravesh Koirala** in partial fulfillment of the requirement for the award of the degree of **"Master of Science in Computer System and Knowledge Engineering"** has been accepted as a bonafide record of work independently carried out by him in the department.

_____

Dr. Dibakar Raj Pant
Head of Department,
Department of Electronics and Computer Engineering,
Central Campus,
Institute of Engineering,
Tribhuvan University,
Pulchowk, Nepal.

Acknowledgement

I would like to express my sincere thanks to Dr. Aman Shakya for his dutiful supervision as well as an active advisory participation during my topic selection and thesis completion period. I would also like to thank Dr. Bal Krishna Bal for his invaluable guidance and advice. In addition, I would like to acknowledge Mr. Ram Hari Koirala for his guidance in Nepali linguistics and Dr. Nobal Bikram Niraula for his support and advice.

I would also like to thank Professor Dr. Shashidhar Ram Joshi, Professor Dr. Subarna Shakya, and Dr. Sanjeeb Prasad Pandey for their encouragement and insights. I am equally grateful to Dr. Dibakar Raj Panta, Head of Department and faculty of the department of Electronics and Computer Engineering for their support. I also heartily thank my family and my classmates for their continued encouragement.

Pravesh Koirala
072MSCS660

Abstract

Stemming is an integral part of Natural Language Processing. It's a preprocessing step in almost every NLP application. Arguably, the most important usage of stemming is in Information Retrieval. While there has been lots of work done on stemming in languages like English, Nepali stemming has only a few mentionable works. This study focuses on creating a Rule Based stemmer for Nepali text. Specifically, it is a affix stripping system that identifies two different types of suffixes in Nepali grammar and strips them separately. Only a single negativity prefix न is identified and stripped. This study focuses on a number of techniques like exception word identification, morphological normalization, word transformation and stemming limit enforcement to increase stemming performance. The stemmer is also tested intrinsically using Paice's method and extrinsically on a basic tf-idf based IR system. Upon testing, the under-stemming error was found to be 5.27% and the over-stemming error was found to be 0.2% which is a superior performance than existing works. The IR was tested on stemmed vs non-stemmed documents and queries using 14 queries and it was found that the stemming scheme increased the average relevance of retrieved documents by 18.6%.

Keywords: Nepali, Stemming, Over-Stemming, Under-Stemming, IR, tf-idf, Paice method

Table of Contents

List of Figures

List of Tables

List of Abbreviations

FST	Finite State Transducer
DMT	Desired Merge Total
DNT	Desired Non-Merge Total
UMT	Unachieved Merge Total
GDMT	Global Desired Merge Total
GDNT	Global Desired Non-Merge Total
GUMT	Global Unachieved Merge Total
GWMT	Global Wrongly Merged Total
IR	Information Retrieval
MT	Machine Translation
NLP	Natural Language Processing
OI	Over-Stemming Index
SW	Stemming Weight
UI	Under-Stemming Index
WMT	Wrongly Merged Total

1. Introduction

1.1 Background and Motivation

Stemming refers to the reduction of a given word into its stem which need not be the morphological root of the word. This is done to reduce the inflection of any particular word into a base form. For example: cats is the inflected form of cat and stemming strips the plurality suffix -s from cats to give cat. Similarly, various tense inflections of the verb *go* like go, going, gone are stemmed to a same stem i.e. go. The straightforward challenge in word stemming is to identify the correct stem. Extending the previous example, the past tense form of the verb *go* .e. *went* needs to be stemmed to *go*, which might not be trivial.

Various NLP applications use stemming as a pre-processing step, for example: POS Tagging, Machine Translation, Document Clustering etc but arguably the most important role of word stemming is in Information Retrieval (IR). IR is an immensely common and important application of Natural Language Processing. It essentially refers to the retrieval of a particular document from a collection of documents. Arguably, the most important example of IR is search engines. Search Engines index massive collection of documents on a daily basis and provide a search interface where users can query or search for a specific document. The nature of search can be for the document itself or for any information contained in a document or for any metadata present in the document, images, multimedia etc.

Similarly, in NLP applications like POS Tagging and Document Clustering, Stemming reduces the word space i.e. the number of unique words to consider by reducing multiple word inflections into a single stem. This not only improves the efficiency of the program by speeding up the execution and reducing memory requirements but also the accuracy by reducing the noise in the dataset.

Stemming need not produce a morphologically correct word. Its only requirement is that it map various word inflections to a common stem even if the stem is not a linguistically correct word. For example, for the word inflections *rider* and *riding*, it suffices to map them both to a common stem (*rid* in this case) instead of their base word (*ride*). In contrast, Lemmatization is a morphological process where words are conflated into their original form or lemmas. For example, stemming *went* might produce *wen* or *went* but Lemmatization yields *go* as the correct lemma. Lemmatization, in general, is a more rigorous process than Stemming and the latter is preferred due to its efficiency.

There are two major problems while stemming: Over-stemming and Under-stemming. Over-stemming is when two separate inflected words are reduced to a same word stem. This is a false-positive when considering the domain of Information Retrieval, since it leads the IR engine to fetch documents which might not contain the search query. Similarly, Under-stemming is when two same inflections of a word are not reduced to a same word stem. This is false-negative. It leads an IR engine to not find documents having a related word inflection. For example: reducing “universe” and “university” to “univers” is over-stemming because the two words are unrelated in modern context. Similarly reducing “alumnus” to “alumnu” and “alumni” to “alum” is under-stemming because both of the words are inflections of same root.

There are various techniques for Stemming. Perhaps the most naive out of them is *Lookup Tables.* These tables have a many to one structure where multiple words can point to the same word stem. While easy to implement, the tables eventually become extensive and the process of mapping a word to stem is manual and particularly tedious. Another technique, which is quite popular, is affix stripping technique where an exhaustive table need not be computed but a very small set of affix stripping rules are used to strip the inflections from the word to get stem.

In a language like Nepali, where most of the words are inflected by using suffixes, affix stripping simplifies to suffix stripping. Constructing a database of known

suffixes, we can apply suffix stripping to derive a stem from the word inflection. While this works in most cases, there might be some exceptions. Consider for example the words *सङ्गीत* and *साङ्गीतिक*. These two are related word inflections but suffix stripping alone is unable to reduce them to the same stem.

There is also a need to identify whether a linguistic entity attached at the end of the word is actually a suffix attaching itself to a base word or is actually a part of the word itself. For instance, in the word "काले" the entity "ले" is actually the part of the word itself whereas in the word "कालेले" the rightmost "ले" is a postpositional suffix. It is imperative to accurately identify when and when not to strip a given suffix because unnecessary stripping leads to over-stemming.

Another challenge in suffix stripping is the difference in writing. For example, both of the word form साङ्केतिक and साङ्केतीक are used interchangeable informally. Unless an assumption about strictness of the writing rules, there is a need to include both of the suffixes *िक* and *ीक*. Not only that, several postpositions can be joined together as in उनीहरुको which contains two postpositions compounded together. To deal with these scenarios, there is a need to repeatedly apply the stripping rules. However, this repetition increases the chance of over-stripping.

Similarly, suffix stripping also requires contextual awareness. For instance, the word बाले can be interpreted in two different ways depending upon the context. It can mean बा + ले *(father did)* or बाले *(lit something on fire)*. Similar is with the word गाउँले, depending upon context, it can either mean *a villager* or *a village did*.

1.2 Problem Statement

Multiple stemming algorithm exists for the English language. Some of which are:

1) Krovetz Stemming Algorithm(1993);
2) Paice/Husk Stemming Algorithm(1990);
3) Porter Stemming Algorithm(1980);

4) Dawson Stemming Algorithm(1974);

For Nepali language, however, a limited work has been done in Morphological Analysis; particularly stemming. The fact that Nepali is an inherently complex language further makes it inaccessible to many analysis. Various derivational and inflectional techniques exist in Nepali grammar which creates plethora of frequently used words in everyday life. For instance, inflection alone is categorized as being of ten types. These inflections can alter a word's structure based on cases such as gender, cardinality, respect, tense (काल) and its aspects (रुप). Moreover, inflections are also based on moods (भाव), voice, causality and negation [6][7].

These inflections make it hard to devise a proper stemming algorithm for Nepali language. In absence of a stemmer, various NLP applications for Nepali which require stemming as a pre-processing step have either not been possible or their implementations have been unsatisfactory. IR for Nepali language has also been pushed back precisely because of the lack of a proper stemming algorithm.

1.3 Objectives

- To devise a rule-based stemmer for Nepali language.
- To evaluate its performance intrinsically based on Over-Stemming and Under-Stemming metrics and extrinsically on a basic IR system

1.4 Scope of Applications

As stated earlier, stemming is a pre-processing step in multiple NLP applications. This work can have significant impact on applications such as IR, Machine Translations, Semantic Analysis, Document Clustering etc.

2. Literature Review

2.1 Previous works in Nepali Language

There have been a few works in Nepali for Morphological Analysis and Stemming. Prasain devised a theoretical model for computational analysis of nepali morphology [1]. In his study, he primarily concerned himself with different morphological categories and processes in Nepali language and the rules involved in deriving these categories. He also focused on developing a computational model i.e. FST for the Nepali morphology. He deals with the morphology of nominals, verbs, adverbs, adjectives, post-positions, case markers, particles, and interjections and also analyses their derivational aspect. However, his works has been all theoretical and no implementation has been done by anyone.

Sitaula proposed a hybrid nepali stemming algorithm which uses affix stripping in conjunction with a string similarity function and reports a recall rate of 72.1% on 1200 words [8]. He has taken into consideration a total of 150 suffixes and around 35 prefixes. After incrementally stripping affixes, Sitaula uses a string similarity method to find the word from Nepali lexicon that is most similar to the stripped stem.

Paul et. al. describes an affix removal stemming algorithm for Nepali text. The system has a database of 120 suffixes and 25 prefixes and a root lexicon of over 1000 words and reports an overall recall accuracy of 90.48% [9].

Shrestha et. al. describes a stemming algorithm for Nepali in which he lists 126 suffix stripping rules. These rules are categorized into three distinct categories. The algorithm strips the suffixes after determining the category of the suffixes. He reports an accuracy of about 88.78% on a total 5000 words [10].

A functional stemming algorithm has been created by Bal Krishna et al [2]. They have developed a database of word-breaking rules for different kind of affixes. The rules are of the form:

एको | 5
नु | 2

Where the first part of the rule statement denotes an affix and the second part (after the pipe) denotes its corresponding entry in the word breaking rule table. They have also maintained a free morpheme list of the form:

कलम | NN
खा | VV
मीठो | ADQ

Here the first part before the pipe is a free nepali morpheme while the second part after the pipe is the POS of the word. Their system outputs the morphology of the word as in the suffixes, root and prefixes. They, however, have not reported the accuracy of their system.

Shrestha et. al. does a comparative study on existing stemmers for Nepali text [11]. The study reports three kinds of stemming algorithms implemented by various studies and identifies them as

1. Rule Based
2. Affix stripping, and
3. Hybrid

The study implements the stemmers of different kinds and tests them against four separate test sets. Affix stripping algorithms outperformed others in the study.

2.2 Previous works in Nepali-like languages

A hindi stemmer was devised by Ramanathan et. al [3] where he first uses a transliteration scheme to transliterate devanagari to ascii. They have maintained a suffix list which is used to strip the word by using the process of longest match. Upon testing the algorithm in 35977 words, 4.6% words were found to be under-stemmed while 13.8% were found to be over-stemmed. They mention that the same algorithm can be successfully used for similar language such as Marathi and Nepali.

An Urdu stemmer is also written by Kansal et. al [4] which uses the rule based approach to stem Urdu words. They report 85.14% accuracy on more than 20,000 words.

Majgaonkar et.al wrote an unsupervised stemmer for Marathi language which basically learned the stemming rules from the given Marathi corpus. They achieved an accuracy of 82.5% on a manually stemmed test data of 1500 words [5].

3. Methodology

3.1 Stemming

Stemming can roughly be divided into 3 types:

- Rule based stemming
- Statistical stemming
- Hybrid stemming

Rule Based Stemming has been known to give above average results in languages like Hindi and Urdu which are morphologically quite similar to Nepali. A specific kind of Rule Based Stemming is Affix removal where the affixes that inflect a given word is stripped to derive the word stem. In Nepali language, prefixes are not known to cause inflection in a word, thus, this study focuses only on suffix removal with the exception of one negativity prefix न.

3.1.1 Suffix Removal

Rule Based stemming maintains a stemming rule list which defines how to strip prefix or suffix from any word. [2], [3] and [4] are all examples of a rule based stemmer. This stemming scheme is especially useful if the list of possible suffixes and prefixes are predefined such as in Nepali language. In many cases, an exception word list is also defined. This word list maintains the list of free morphemes which should not be stemmed. This helps in reducing over-stemming errors. [3], [5], [10] are an example of Rule Based Stemming where only a suffix list is defined.
A typical Rule based stemming flowchart looks as follows:

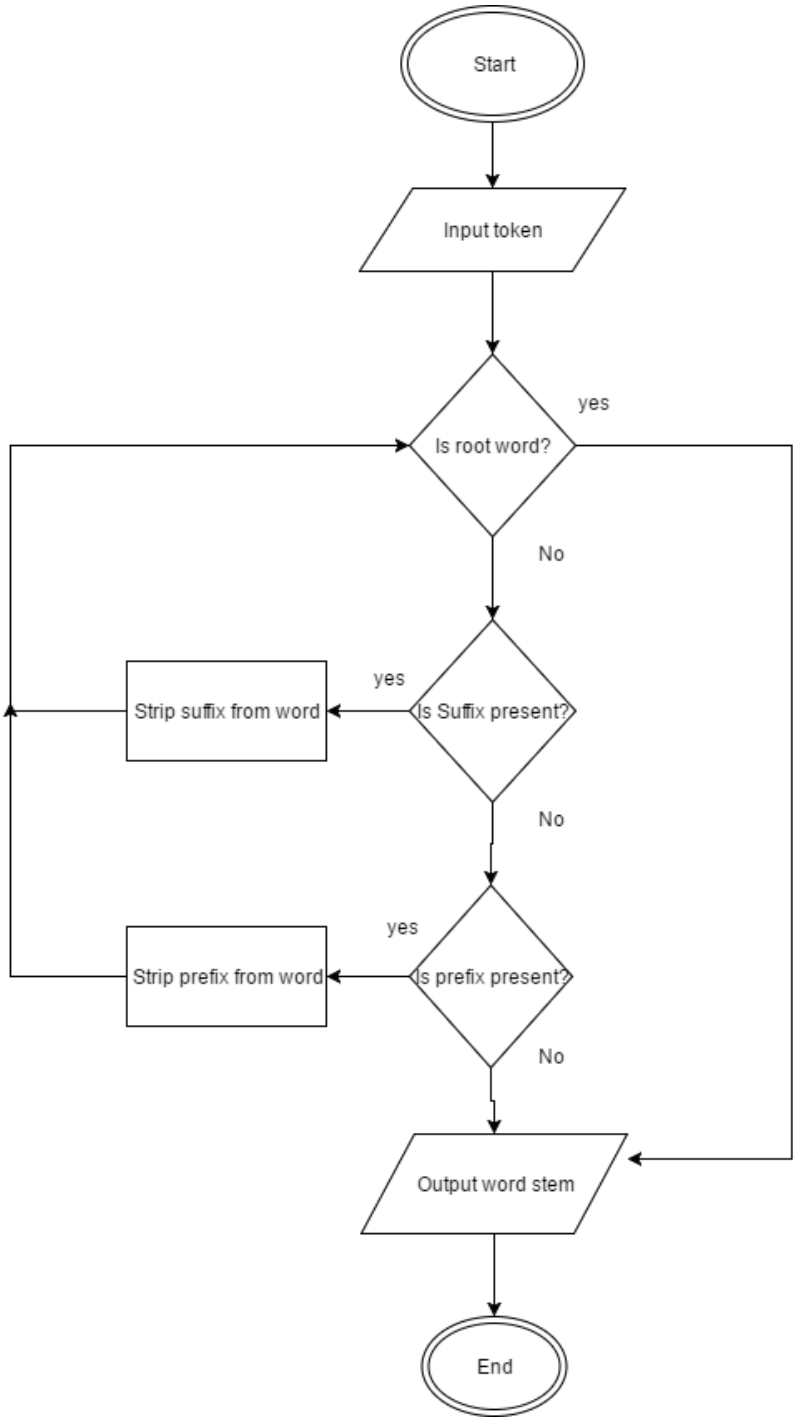

Figure 1: Flowchart for a typical Rule based stemmer.

3.2 Morphological Normalization

Among the vowels present in Nepali language, the vowel pairs <इ, ई> and <उ, ऊ> in both their dependant and independant forms are often confused while writing. Same is the case with some of the consonant groups like <व, ब>. To make the stemmer more robust to these common grammatical errors, a morphological normalization

scheme is introduced where the often confused vowels and consonants are normalized into a single entity. Concretely, all occurance of the vowel ई are replaced with इ and so on. A more detailed normalization scheme is outlined below.

Table 1 : Morphological normalization scheme

Vowel / Consonant	Normalized To
ई	इ
ी	ि
ऊ	उ
ू	ु
व	ब
श	स
ष	स
ँ	*Nil* (all occurances removed)

All of the inputs to the stemmer are morphologically normalized during the stemming process. This includes any input word to be stemmed, the suffix or prefix list, and the word exception list; which is to say; the stemmer only deals with normalized Nepali words.

3.3 Suffixes

Suffixes are linguistic entities that inflect a word by attaching themselves to the end of it. For example, the plural marker हरु, inflects the singular word केटा into केटाहरु. In Nepali, there are many suffixes, but these have been broadly divided into two categories in this study.

3.3.1 Type I Suffixes

These suffixes are agglutinative and called postpositions (विभक्ति in Nepali). Their grammatical function is identical to the prepositions in English language. They attach themselves to mostly Nouns and Pronouns but are not in any way restricted to the two parts of speech. Some examples of these suffixes are:

- ले
- लाई
- बाट, द्वारा
- लागि, निम्ति
- देखि, बाट
- को, का, की
- मा
- अघि
- पछि
- पहिले
- पश्चात
- माथि
- मुनि
- भित्र
- बाहिर
- वारि
- पारि
- झैँ
- जस्तै

A total of 85 Type I suffixes are identified in this study (see Appendix A).

3.3.2 Type II Suffixes

These suffixes, also called *प्रत्यय* in Nepali mostly inflect verbs. Some of the identified suffixes of this type are:

- छ
- ँछ
- ा
- ेका
- एका
- न
- छु
- ँछु

- छे
- छौ
- ने
- इने
- यो
- दा
- दै
- ँदै
- नु
- ए
- एँ
- े
- ँ

It is worth noting that both the suffixes एका and ेका are included in the list even though they are essentially identical grammatical elements and only differ in respect to whether the leading vowel is independent or is dependent to the preceding consonant. A total of 161 of these type II suffixes were identified (see Appendix B).

3.4 Suffix Stripping Rules

Suffix stripping is done on the basis of the type of the suffix i.e. type I and type II suffixes are stripped separately.

3.4.1 Stripping Type I suffixes

Stripping these suffixes is a non-trivial process. This can be attributed to two major facts:

Firstly, identification of these suffix is challenging. As was discussed earlier, some of these suffixes occur as a part of word itself. For instance, the word नेहरु is the name of a reputed Indian politician and not the suffix हरु attached to the base ने. There are many more examples of such exception words. Before stripping type I suffixes, an extensive exception word list has to be created.

There are essentially two ways in which the exception word list could be created:

- Lexicon Based
- Corpus Based

In Lexicon based approach, the entire contemporary Nepali Lexicon is taken as the exception list. The lexicon contains words that are root on their own. Words like मामा, पहिले, नलागि etc are already listed on the lexicon. On the other hand, proper nouns, abbreviations etc are not. So, the stemmer won't recognize the word उमा or ओबामा as exceptions.

In Corpus based approach, all the words present in the corpus that have the suffixes/prefixes of interests are listed out. Out of those words, the words that are exceptions are manually identified. This approach can identify words that occur in the corpus but not in a standard Nepali lexicon.

In the study, corpus-based approach is used for all words ending with suffix of interest and occurring a minimum number of time (threshold) inside the corpus. Thresholding is done to reduce the number of words to deal with. For the purpose of this work, the threshold was chosen to be 10. A total of 181 of these exceptions words were identified (see Appendix C).

Another challenge in stripping type I suffix is that these suffixes can be chained together i.e. the word उनीहरुलाई is a word created by chaining two different type I suffixes. This requires repetitive stripping of the suffixes while checking the intermediate results against the exception word list.

The algorithm used to strip type I suffix can be outlined as follows:

Procedure:

1. *Read suffixes, and exception words*
2. *Take input sentence*
3. *Tokenize the sentence into tokens using space and punctuation as word boundary.*
4. *For each word:*
    a. *Check if word is in exception list*
        i. *If yes, do not tokenize, continue the loop*
        ii. *If no, proceed to tokenization*

b. Separate the word into root and suffix.

c. Repeat steps a..b for the newly tokenized root.

5. Output list of tokens

3.4.2 Stripping Type II Suffixes

Stemming these suffixes are particularly tricky largely due to the inherent structure of Nepali Morphology. For example, consider the suffix छ. When appended with the root जा, the combination introduces a new sound of न.

जा + छ = जान्छ (Introduction of a न in the middle)

However, no such phenomenon is observed when the root is गर् (do)

गर् + छ = गर्छ

Similarly, the suffix यो, when used with the root जा, changes the morphology and phonetics but it doesn't do the same for the root खा or गर्

जा + यो = गयो (change of the word structure)

खा + यो = खायो

सुत् + यो = सुत्यो

These variations are not only observed in verb but also in noun roots. For instance, the इक suffix is known to change the morphology of nouns in the following way:

सङ्गीत + इक = साङ्गीतिक

समाज + इक = सामाजिक

भुगोल + इक = भौगोलिक

I.e. change of the dependent vowels (अ to आ) at the start of the word.

To take these factors into consideration, we introduce a word transformation rule. In simple terms, if the word contains the इक prefix, the dependant vowel at the start of the word is changed accordingly. The vowel आ becomes अ, vowel औ becomes उ and the vowel ऐ becomes इ. Using this transformative rule, the word नैतिक would be transformed to the word नितिक. It is important to observe that this map does not map a word to its stem, rather only to an intermediate word, which will be then further

processed to produce the correct stem. The intermediate word might not be grammatically correct one. The rationale being that the word नितिक and the word नीति would conflate to the same once they are morphologically normalized and then stemmed.

The stemming algorithm in itself is quite simple. In fact, after taking into account the variations in word morphology by addition of suffixes, the rest of the process is the repeated stripping of the suffixes in a longest suffix first approach. This stripping is done until further stripping is not possible. In the event that any particular stripping rule decreases the word size to below a set threshold, that rule is discarded. This is done to prevent over-stemming of the word. The threshold value for this project was taken 2 by observing the error rates as per the testing method described in 4.3.1.

The stripping algorithm of type II suffixes is as follows:

Procedure:

1. *Read suffixes, and words transformation map.*
2. *Take input tokens*
3. *For each word:*
    a. *Check if word has इक suffix*
        i. *If yes, transform the word according to rule described in 3.2.2*
        ii. *If no, proceed.*
    b. *For each suffix in the suffix list from longest to shortest suffix*
        i. *If the suffix is present in word*
            1. *If stripping leads to word length above the threshold, strip it.*
            2. *Else ignore*
            3. *Continue the loop in (3b)*
    c. *If no suffix could be stripped, break the loop else repeat step 3b again.*
4. *Output list of tokens*

3.4.3 Prefix Stripping

Only one kind of prefix has been stripped in this work, the negativity prefix न. As the name implies this prefix is a verb inflection that negates it. For instance, the verb आएको (has come) can be inflected to नआएको (has not come) by addition of the prefix. The rule for stripping this prefix is quite straightforward i.e. it is stripped if it occurs at the beginning of the word.

3.5 Stemmer

A simple outline of the built system is shown below:

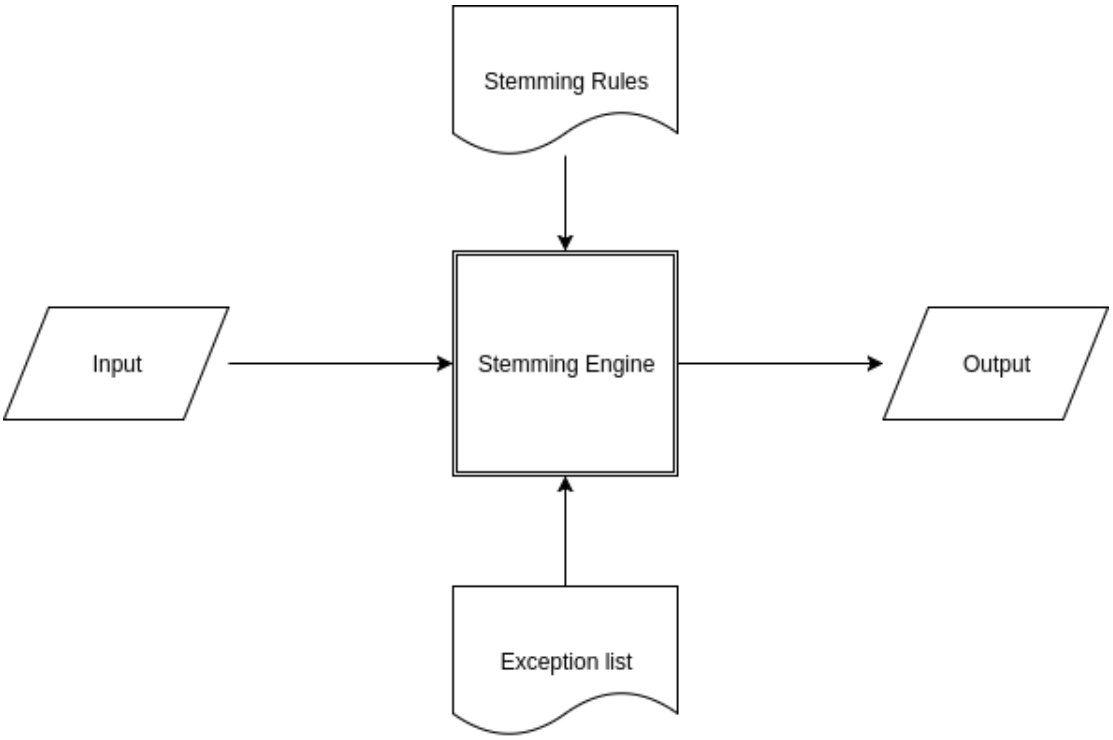

Figure 2: Block diagram of a Rule Based Stemmer

3.5.1 Input

Input is typically a Nepali word or a string of words. An entire text document encoded with UTF-8 encoding can be entered as well. The stemmer morphologically normalizes the input according to the scheme mentioned in 3.2.

3.5.2 Exception list

This is a list of words that are known to be root words i.e. not inflected with any suffix or prefix. For example: घर, माया, नेहरु, नेपाल etc.

For now, these words have been taken from the corpus by the process of manual eyeballing. The list also includes words that are non-native to Nepali language but are root words. An example is ओबामा, which does not exist in Nepali lexicon but is a root word. The exception words are also normalized as per section 3.2.

3.5.3 Stemming Rules

Stemming rules, is the collection of all known type I and type II suffixes. There are 54 type I suffixes whereas 115 type II suffixes. All of these suffixes are normalized.

3.5.5 Stemming Engine

This is the central application that takes the input wordlist, reads rules from the rules base and stems the word. The matching of word with the free morpheme list and the exception list is also handled by this engine. After it stems the word, it outputs the stem of the word. A general flowchart of what this engine looks like is shown below.

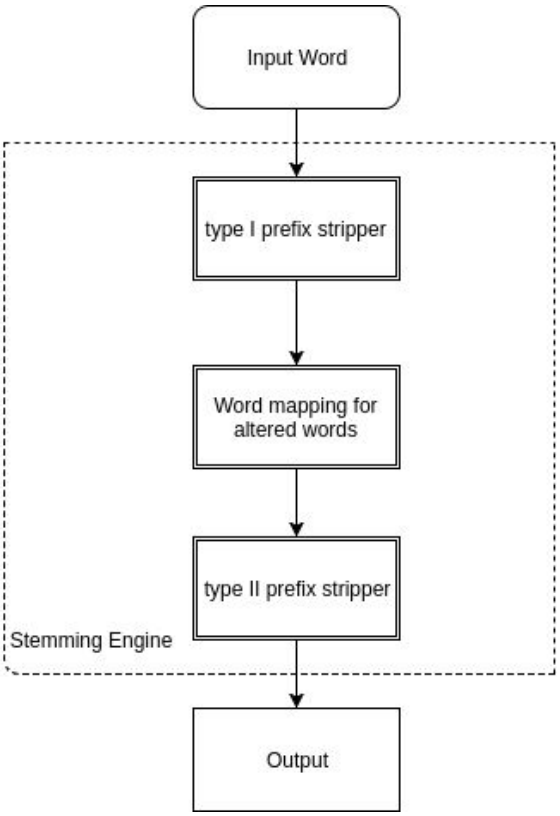

Figure 3: Block diagram of stemming engine.

3.6 Data

To test the stemming rules and evaluate the over/under stemming errors, a corpus was constructed. This corpus was derived from various online news portals such as Setopati, Nagariknews, eKantipur etc. The corpus contained articles from various different areas including news, sports, arts and literature etc. Corpus contained a total of 4387 news articles with the total word count of 1181343 and total unique word count of 118056. Each news article, on average, contained 269 total words and 181 unique words.

3.7 Tools

- Python 2.7
- Pylex library
- Web.py server
- Pycharm IDE

4. Results and Discussion

4.1 Output

The output of the stemmer is essentially the stem of the word, or if it is a wordlist, the stemmed wordlist. A simple GUI is provided to operate the stemmer. The GUI contains two controls. A textbox to input a Nepali word or wordlist, and a button control to stem the contents of the textbox. Once appropriate input is provided and the button is pressed, the GUI generates a table with two columns: the original word and the stemmed word.

A snapshot of the GUI is shown below:

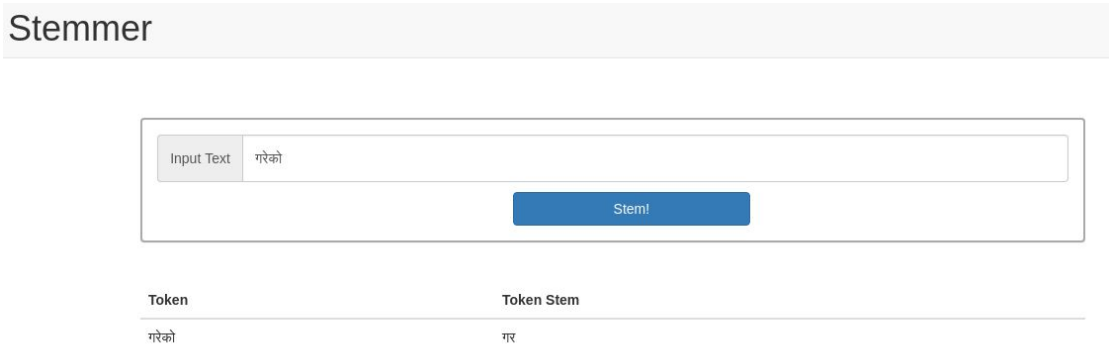

Figure 4: Stemming the word "गरेको"

4.2 Performance Evaluation

Performance evaluation of a stemming algorithm is ideally done on the task that the stemming was done for. For example, if stemming were done for IR purposes, then the evaluation of stemming would be on the basis of accuracy or recall of the IR system. This is also known as Extrinsic Evaluation. In contrast, Intrinsic Evaluation is where the stemmer is tested in itself, without actually using it for any external application.

Both Extrinsic and Intrinsic performance evaluation for this stemmer was done. For Intrinsic evaluation, a testing method known as Paice method [12] was used while for Extrinsic evaluation, a trivial Information Retrieval Engine was constructed.

4.2.1 Paice Method

Paice method for evaluation of stemmers is based on under-stemming and over-stemming errors. In this method, a concept group is first defined where multiple word inflations of a single word-concept are grouped together. To illustrate this concept, consider the following words: walk, walking, go, went, gone, eat.

Among these six words, three concept groups can be defined. Walk and walking would constitute a concept group because both these words are inflations of the verb walk. Similarly, the words go, went, and gone would be mapped to a single concept group because these three words are associated with the verb go. The remaining group would constitute the single word go. After defining the concept groups, Paice method operates by counting the actual over and under stemming errors and calculating certain Performance Indices.

4.2.1.1 Performance Indices

These are the indices used to calculate under-stemming and over-stemming errors. The equations are directly derived from [12]. A Desired Merge Total (DMT) for a single concept group is defined as the number of word pairs that are supposed to

conflate to the same stem. Ideally, it equals the number of word pairs in the concept group. It is calculated as

$$DMT_g = 0.5\, n_g\, (n_g - 1) \qquad 4.1$$

Where n_g is defined as the number of words in the group.

A Desired Non-Merge Total (DNT) for a group is then defined as:

$$DNT_g = 0.5\, n_g (W - n_g) \qquad 4.2$$

Where W is the total number of words being evaluated i.e. sum of words in all concept-groups. This metric is actually the number of possible word pairs between the words of a concept group and the words outside of the concept groups.

By summing these two metrics across all concept-groups, two new indices GDMT (Global Desired Merge Total) and GDNT (Global Desired Non-merge Total) are calculated.

$$GDMT = \sum_g DMT_g \qquad 4.3$$

$$GDNT = \sum_g DNT_g \qquad 4.4$$

After stemming is applied to the text, some groups might have two or more distinct word stems. This is due to the *Under-Stemming* of the words. To quantify these errors, a new metric is used called the Unachieved Merge Total (UMT) for a group. It is defined as:

$$UMT_g = 0.5 \sum_{i=1}^{s} u_i (n_g - u_i) \qquad 4.5$$

UMT is a measure of the number of pairs in a group that did not successfully conflate to the same stem. Summing over UMT across all groups, we get a Global Unachieved Merge Total (GUMT).

$$GUMT = \sum_g UMT_g \qquad 4.6$$

Using GUMT and GDMT, we can calculate a *under-stemming index* (UI) as follows:

$$UI = GUMT / GDMT \qquad 4.7$$

Post-stemming, it can also be found that words of different concept groups conflate to a same stem. This is *Over-Stemming*. To quantify this error, a stem-group is constructed. Stem groups is essentially all the words that conflate to a particular word stem. Any stem-group that consists of words from different concept-groups contains over-stemming errors. Suppose a stem-group has n_s items which are derived from t different word-groups and suppose the numbers of words from the word-groups are $v_1, v_2 .. v_t$. The metric Wrongly Merged Total (WMT) can then be calculated as:

$$WMT_s = \sum_{i=1}^{t} v_i(n_s - v_i) \qquad 4.8$$

WMT_s gives the number of word pairs that wrongly conflated to the single stem. Summing WMT across all the stemming groups, we get the Global Wrongly Merged Total.

$$GWMT = \sum_{s} WMT_s \qquad 4.9$$

Using GWMT and GDNT, we can get a new metric called *over-stemming error* (OI):

$$OI = GWMT / GDNT \qquad 4.10$$

The metrics *OI* and *UI* give a measure of the Over and Under Stemming errors. Using these two, a new metric called a Stemming Weight can be calculated.

$$SW = OI / UI \qquad 4.11$$

The stemming weight index is greater than one if the stemmer has fewer under-stemming errors than over-stemming errors i.e. the stemmer aggressively stems the word. These types of stemmer are called *heavy stemmer*. In contrast, if the number of under-stemming errors is more than the over-stemming errors, the stemmer is less aggressive in stemming. These stemmers are called *light stemmer*.

4.2.1.2 Test Setup and Results

For evaluating the stemmer according to Paice method, 497 concept groups were defined. Each concept groups contained at least two related words with the maximum being thirty-nine words. A total of 1813 words constituted the concept groups. Some examples of the groups are as follows:

- तपाईँ, तपाई, तपाईंको, तपाईहरू, तपाईंले, तपाईको, तपाईले, तपाईहरु
- हुनुपर्ने, हुनु, हुनुपर्छ, हुनुहुन्छ, हुनुहुन्थ्यो
- राख्ने, राख्न, राख्दै, राख्नु, राख्नुपर्ने, राख्दा, राख्नुपर्छ
- मानिस, मानिसको, मानिसहरु, मानिसलाई, मानिसले, मानिसमा, मानिसहरुको, मानिसहरुले

These words were derived from the top 10,000 most frequent words occurring in the corpus described in section 3.6. After running Paice method of evaluation on the stemmer using these concept groups, following results were obtained.

Table 2: Values for various metrics in Paice's method.

Metric	Values
Global Desired Merge Total (GDMT)	8274
Global Unachieved Merge Total (GUMT)	436
Global Desired Non-Merge Total (GDNT)	2742411
Global Wrongly Merged Total (GWMT)	4729

These values can be summarised in the chart below:

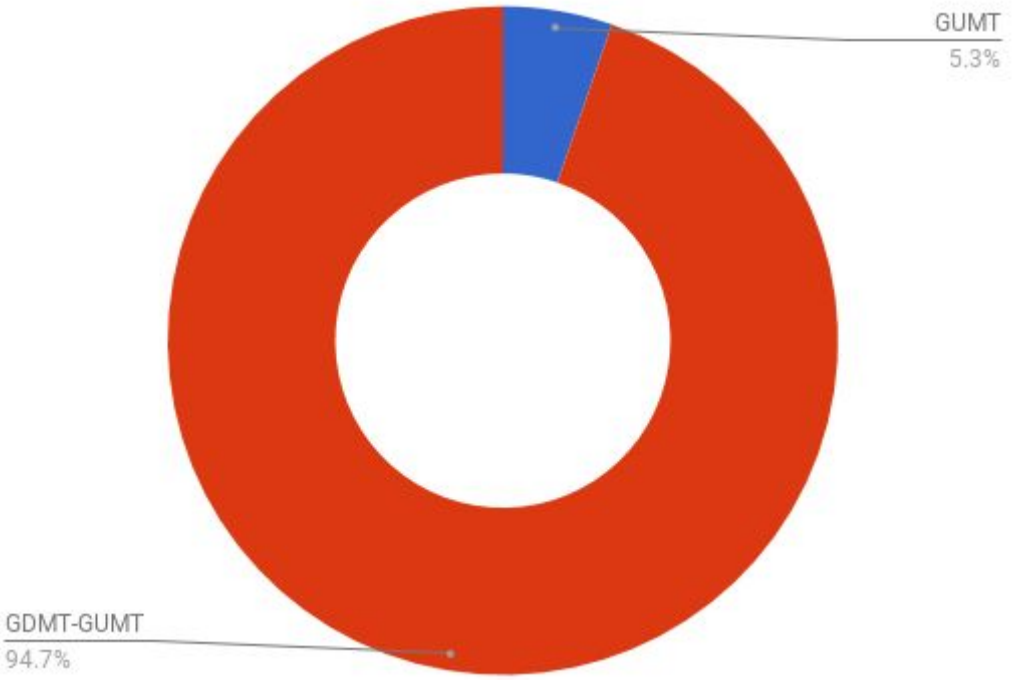

Figure 5: Stemming ideal merge (GDMT) vs unachieved merge (GUMT)

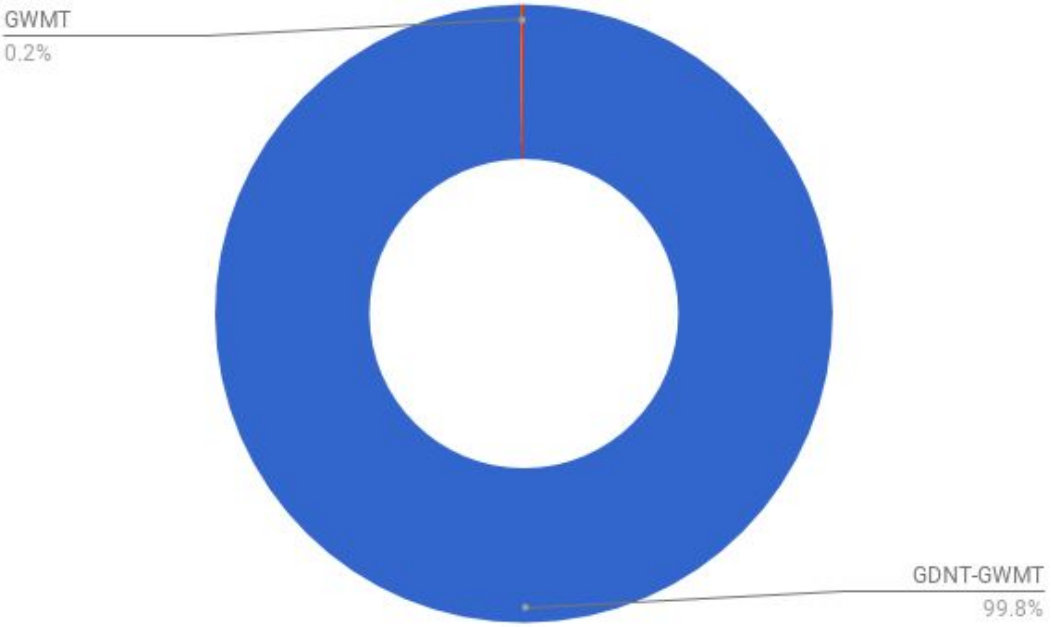

Figure 6: Stemming ideal non-merge (GDNT) vs achieved non-merge (GWMT)

Using the metrics above, the under-stemming index and the over-stemming index can be calculated as per equations 4.7 and 4.10.

Table 3: UI, OI and SW using Paice's Method

Metric	Values
Understemming Index (UI)	0.0527
Overstemming Index (OI)	0.002
Stemmer Weight (SW)	0.038

This shows that the stemmer has high understemming error in contrast to over-stemming error. Which implies that the stemmer is a *light stemmer* i.e. it has a tendency to not strip suffixes aggressively.

4.2.2 Information Retrieval Test

A most accurate and pragmatic test for any Stemmer is to actually implement a NLP application based on that Stemmer and then check for the performance of that application. For the purpose of this thesis, a crude IR system was developed using the Stemmer and then tested on a prepared dataset upon a subset of the corpus described in 4.1.

4.2.2.1 Background

Modern IR systems employ various measures like query expansion (where a simple input query is reconstructed to multiple queries for getting a wider coverage) to sophisticated relevancy algorithm like pagerank. For the purpose of this thesis, however, only a simple IR system has been developed where both documents and queries are modeled using the *bag of words* model and the ranking is done by using *tf-idf* metric which has been shown to give good results for document retrieval [13].

Bag of Words is a simple technique to model a set of documents mathematically. It is widely used in text classification. It is simple and intuitive to use but a drawback of

this model is that we lose the ordering information which also leads in the loss of the semantic information. For example: using *bag of words* model for IR, the two queries *Fire Truck* and *Truck on Fire* return same set of documents even though the two queries are semantically different.

For the purpose of this thesis, a document **d** is defined as a body of text. Similarly, a document collection **D** is the set of all the documents **d**. Mathematically,

$$\mathbf{D} = \{d_0, d_1, \ldots d_S\} \qquad 4.12$$

Where S is the total number of documents in the document collection **D**.

We also define a corpus as the set of all unique words in the documents. For instance, a corpus of size K is defined as:

$$\mathbf{C} = \{w_0, w_1, w_2 \ldots w_K\} \qquad 4.13$$

Where w_0, w_1 … w_K are *unique words* in the document collection **D**.

Using the corpus, the documents that are to be used for IR purposes can be modeled as a *vector*. For a corpus of size *K*, the document vector is of *K* dimension.

This vector is constructed in the following ways. Consider the document *d* of length L.

$$d = w_0 w_1 w_2 \ldots w_L \qquad 4.14$$

Where, w_0, w_1 etc are words in the document which are *not* necessarily unique.

For any given word w_i in the corpus, its non-normalized term frequency in the document *d* is defined as:

$$ntf_{i,d} = N_{i,d} \qquad 4.15$$

Where $N_{i,d}$ is the number of occurence of the word w_i in the document d. Again, the normalized term frequency or simply the term frequency of the document is defined as:

$$tf_{i,d} = ntf_{i,d} / \| d \| \qquad 4.16$$

Where $\| d \|$ is the euclidean norm of the document d given by

$$\| d \| = \sqrt{\sum_{i}^{K} (ntf_{i,d})^2} \qquad 4.17$$

Simply using term frequency as relevance measure would bias the relevance metric to the most frequently occuring words regardless of their relevance to the document. To remedy this, a new metric called the document frequency is used. The document frequency (df) of a word w is defined as:

$$df_w = N_w \qquad 4.18$$

Where N_w is the number of documents containing the word w. Again, the metric inverse document frequency (idf) for a word is calculated as:

$$idf_w = log_2(N / df_w) \qquad 4.19$$

The relevancy metric tf-idf for a given word w_i in a given document d is calculated as:

$$tfidf_{w_i,d} = tf_{w_i,d} * idf_{w_i} \qquad 4.20$$

Using equation 4.20, a column vector for the document described in equation 4.14 can be constructed as follows:

$$V_d = \begin{bmatrix} tfidf_{w_0,d} \\ tfidf_{w_1,d} \\ . \\ . \\ tfidf_{w_U,d} \end{bmatrix} \qquad 4.21$$

V_d is what is called the document vector of document d having U unique words. Once document vector for every document d in the document collection D is defined, relevance score of any document for any given query q can be obtained.

A query q is also a document typically having fewer words than the documents in the document collection. Regardless, it can also be modelled as a vector called the query vector. The query vector is defined as follows:

$$V_q = \begin{bmatrix} tfidf_{w_0,q} \\ tfidf_{w_1,q} \\ . \\ . \\ tfidf_{w_Q,q} \end{bmatrix} \qquad 4.22$$

V_q is the query vector for the query having Q unique words. The relevance of any document Vd and any query Vq can be calculated by the cosine similarity of the vectors. Mathematically,

$$rel(V_d, V_q) = V_d \,.\, V_q \,/\, (\|V_q\| \,.\, \|V_d\|) \qquad 4.23$$

Where $V_q \,.\, V_d$ is the dot product of the two vectors and $\| V_q \|$ and $\| V_d \|$ are the euclidean norm of the vectors V_q and V_d respectively as defined in equation *4.17.*

4.2.2.2 Test Setup and Results

For the purpose of this test, total 100 documents were sampled from the corpus in 4.1. Then, 14 queries were constructed for retrieval. These queries contained one to three words and were constructed manually using the gathered documents. Some of the queries are shown below:

- पोखरीमा विष
- साझा बस
- कतार राजदुत
- अखिल क्रान्तिकारी

Using the TF-IDF ranking scheme mentioned in 4.2.2.1, two independent information retrieval experiment were carried out for each query. The first experiment was done without stemming the documents or queries while the second experiment was done on the stemmed document and queries. The topmost result of the information retrieval i.e. the document with the highest relevance score for the given query for both experiments were taken and 3 native nepalese human judges were asked to assess the relevance of the retrieved document on the scale of 1 to 5, 1 being the least relevant while 5 being most. If the query failed to return any document in any experiment, the relevance was assumed to be 0.

A summary sheet of the experiment is shown below. The form used to collect data could be found at *https://goo.gl/forms/Kt82ZTzFeW37VMVC3*.

Table 4: Summary sheet for IR relevance scoring using vs not using stemmer

Query	Type	Participant 1	Participant 2	Participant 3	Average	Difference
पोखरीमा विष	*unstemmed*	4	4	4	4	0
	stemmed	4	4	4	4	
साझा बस	*unstemmed*	3	4	5	4	0
	stemmed	3	4	5	4	

कतार राजदुत	*unstemmed*	1	1	2	1.33	2
	stemmed	3	3	4	3.33	
अखिल क्रान्तिकारी	*unstemmed*	4	5	5	4.67	0
	stemmed	4	5	5	4.67	
भलिवल प्रतियोगिता	*unstemmed*	5	4	5	4.67	0
	stemmed	5	4	5	4.67	
भक्तपुर मन्दिर	*unstemmed*	4	5	4	4.33	0
	stemmed	4	5	4	4.33	
कृषि व्यवसाय	*unstemmed*	2	1	2	1.67	-0.67
	stemmed	1	1	1	1	
नेपालको राष्ट्रपति	*unstemmed*	4	5	5	4.67	-1.67
	stemmed	4	1	4	3	
नेपाली कांग्रेस नियमावली	*unstemmed*	1	1	1	1	2.33
	stemmed	3	3	4	3.33	
सिंहनाथ गण	*unstemmed*	4	5	4	4.33	0
	stemmed	4	5	4	4.33	
राजेश हमालको आमा	*unstemmed*	1	1	1	1	3.67
	stemmed	4	5	5	4.67	
दशैँ	*unstemmed*	5	3	5	4.33	0
	stemmed	5	3	5	4.33	
विदेशमा रोजगारी	*unstemmed*	1	1	2	1.33	2.67
	stemmed	3	4	5	4	
दसैँ	*unstemmed*	0*	0*	0*	0	4.67
	stemmed	4	5	5	4.67	

Scores with * indicate that the document was not found for that experiment, thus, relevance was assigned to be 0. The difference in average relevance score of the retrieved document with stemming and without stemming was calculated for each query and the differences were averaged at the end. The average gain in the relevance was found to be 0.93. The chart below compares the average relevance gain for individual queries.

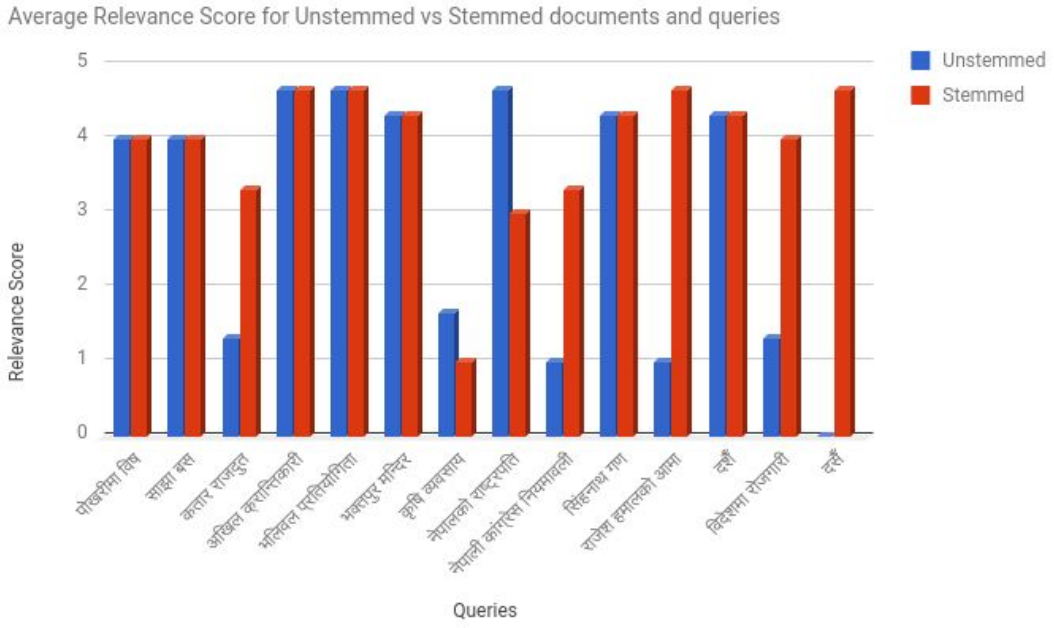

Figure 7: Average relevance score comparison for queries using stemming vs not raw document (unstemmed)

5. Conclusions

A rule based suffix stripping stemmer for Nepali was created in this project by identifying multiple suffixes in Nepali language. These suffixes were categorized into two broad types: type I which primarily consisted of postpositions and type II which consisted of case markers. Both intrinsic and extrinsic evaluation was then performed on the stemmer. The intrinsic evaluation was done using Paice's method. The extrinsic evaluation was done on a basic tf-idf based information retrieval engine where sample documents were queried against a manually constructed query set using both stemmed and non-stemmed documents. The results found were as follows:

1. Intrinsic Evaluation with Paice's method gave a under-stemming error of 5.27% and over-stemming error of 0.2% implying that the stemmer is a *light stemmer*
2. The average gain in relevancy of retrieved document for the IR was found to be 0.93 i.e about 18.6%.

6. Limitations and Future Works

The limitations of the work can be summarized as follows:

1. Apart from the negativity prefix न, other prefixes have not been considered for stripping. Prefixes can be included to improve upon this work.
2. Too few words have been tested during the intrinsic evaluation in section 4.2.1, an exhaustive number of test words can be constructed to evaluate the stemmer performance.
3. The number of searchable documents and the queries are few for the extrinsic evaluation in 4.2.2. More documents can be used with many queries to evaluate extrinsic performance.
4. Context-sensitive words like बाले, गाउँले etc have not been properly considered. This work could be improved upon to consider the context of the words as well.
5. The IR for section 4.2.2 is a trivial one that only uses tf-idf metric for similarity. A more robust IR system can be constructed to check for the performance of stemmer.
6. Probabilistic stemming for Nepali can be worked upon and compared against this work i.e. Rule based stemmer.

7. References

[1] Prasain, Balaram. A computational analysis of Nepali morphology: A model for natural language Processing. Diss. Tribhuvan University, 2011.

[2] Bal, Bal Krishna, and Prajol Shrestha. "A Morphological Analyzer and a stemmer for Nepali." PAN Localization, Working Papers 2007 (2004): 324-31.

[3] Ramanathan, Ananthakrishnan, and Durgesh D. Rao. "A lightweight stemmer for Hindi." the Proceedings of EACL. 2003.

[4] Lehal, Rohit Kansal Vishal Goyal GS. "Rule Based Urdu Stemmer." 24th International Conference on Computational Linguistics. Vol. 267. 2012.

[5] M. Majgaonker, Mudassar & Siddiqui. Discovering suffixes: A Case Study for Marathi Language. International Journal on Computer Science and Engineering (2010).

[6] Mathew, D. A Course in Nepali, RatnaPustak Bhandar, 1998

[7] Adhikari, H. R, Bhandar, B.P and Bhotahiti, Samasamayik Nepali Vyakaran, Kathmandu, Third Edition 2062 B.S

[8] Sitaula, Chiranjibi. "A hybrid algorithm for stemming of Nepali text." Intelligent Information Management 5.04 (2013): 136.

[9] Paul, Abhijit, Arindam Dey, and Bipul Syam Purkayastha. "An Affix Removal Stemmer for Natural Language Text in Nepali." International Journal of Computer Applications 91.6 (2014).

[10] Shrestha, Ingroj, and Shreeya Singh Dhakal. "A new stemmer for Nepali language." Advances in Computing, Communication, & Automation (ICACCA)(Fall), International Conference on. IEEE, 2016.

[11] Shrestha, Ingroj, Shreeya Singh Dhakal, and Madan Kadariya. "A Comparative Study of Stemming Algorithms for Nepali Language." National Students' Conference on Information Technology (2016):

[12] 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. Springer-Verlag New York, Inc., 1994.

[13] Ramos, Juan. "Using tf-idf to determine word relevance in document queries." Proceedings of the first instructional conference on machine learning. Vol. 242. 2003.

Appendix A

List of Type I suffixes.

मा	बमोजिम	भन्दा
बाट	निम्ति	यता
ले	निमित्त	ग्रस्त
लाई	लागि	बेला
द्वारा	सम्म	कारण
लागि	पर्यन्त	गरि
निम्ति	मात्र	मै
देखि	केवल	समक्ष
बाट	तिर	बिच
को	निर	रत
का	सामुन्ने	कारी
की	तर्फ	लगायत
हरु	पट्टि	अन्तर्गत
अघि	सित	भयो
पछि	सँग	पनि
पहिले	समेत	पूर्ण
पश्चात	सहित	अगाडि
माथि	सट्टा	पछाडि
मुनि	विना	स्तरीय
भित्र	बाहेक	बद्ध
बाहिर	पटक	बित्तिकै
वारि	मध्ये	हुन्थ्यो
पारि	स्थित	होला
झैँ	विरुद्ध	पर्दछ
झैं	पालिका	परेको
जस्तै	मध्ये	हुन्छ
समान	बारे	कै
सदृश	लगत्तै	कर्मी
		सँगै

Appendix B

List of Type II suffixes.

न्छ्यौ	ाएको	ेस्
पूर्वक	थ्यौ	ुन्
ुन्जेल	ुँला	ँदै
न्छेस्	लान्	सार
उन्जेल	दिने	ोस्
न्छिन्	इसके	न्छ
इन्जेल	औंला	इस्
िन्जेल	छिन्	िया
अनुसार	ोभित	छौँ
ुपर्ने	इन्छ	उन्
चाँहि	लास्	आएर
रहेकी	ौंला	इने
रहेको	िन्छ	ोभन
रहेका	ियाँ	एको
तुल्य	छेस्	छन्
थ्यौँ	डालु	एकी
ात्मक	भएको	ेका
चाहीँ	भएका	ेको
खेरिन	ँछन्	ाडी
न्छस्	पर्छ	एका
ोत्तर	छ्यौ	ेकि
मात्र	आएका	छस्
न्छन्	आएको	एन्
ोत्तम	ाउनु	शील
लिस्	स्थल	नन्
थिन्	ौँला	िस्
न्छौ	भरि	यौं
सुकै	्नो	ँछु
िसके	ेलि	नुस
आउनु	थैं	इया
लिन्	ओस्	ईन्
होस्	ेन्	झैँ
ीकरण	ालु	ाएर
इयाँ	होस	ीन्
यालु	साथ	भर
खेरि	िलो	ेर
उँला	एस्	नु

ने
ना
थे
सक
आइ
आउ
आए
दै
दो
दा
छु
छे
उँ
ँछ
छौ
ीय
तै

औं
ाइ
ाए
ता
एर
एँ
इत
इन
यौ
यो
गत
कै
ित
ौं
िक
ं
ि

ा
ू
क
ो
ु
ँ
ई
न
ी
ौ
ए
ओ
छ
ै
ु

Appendix C

List of 181 identified exception words for Type I suffixes

आमा	डिस्प्ले	हल्का
शर्मा	ढिलाई	गायिका
जम्मा	बोलाई	शिक्षिका
लामा	आगलागि	कार्यतालिका
जिम्मा	नलागि	तालिका
सिनेमा	अर्को	काका
सीमा	चर्को	हलुका
बिमा	निको	लुका
उमा	अरनिको	धक्का
मामा	पोको	ईलाका
विश्वकर्मा	ढिस्को	धोका
सिमा	डाँको	भुमिका
बीमा	अर्का	शङ्का
पश्चिमा	ठेक्का	लङ्का
रमा	खड्का	अम्बिका
क्षमा	मौका	जानुका
बर्मा	अमेरिका	हरितालिका
निमा	शंका	रोक्का
गोमा	इलाका	एकताका
अहिले	ढोका	पोका
एमाले	आशंका	कालिका
पहिले	उपत्यका	झट्का
कहिले	पक्का	भाका
थाले	टिका	ताका
वाग्ले	बेलुका	हिंसाका
माले	बालबालिका	प्रवेशिका
घले	बालिका	अफ्रिका
काफ्ले	टीका	राधिका
स्कुले	तरिका	बाँकी
तैले	निर्देशिका	कार्की
बोले	हेक्का	अमेरिकी
आले	नगरपालिका	धम्की
खोले	चौका	पक्की
हाले	नाका	कास्की
खेले	खाका	बेलुकी
काले	श्रीलंका	चौकी

अर्की
अमेरीकी
बुढाथोकी
टर्की
सार्की
कलंकी
अफ्रिकी
नसकी
गण्डकी
जानकी
हुलाकी
रोकी
दैनिकी
मुलुकी
कलङ्की
सुराकी
तोकी
झाँकी
बोकी
रुघाखोकी
कालीगण्डकी
काकी
लक्की
मानविकी

ट्याङ्की
घुर्की
पक्कापक्की
चर्काचर्की
ढुकढुकी
पेश्की
चुस्की
वाँकी
सकी
वाकिटकी
डिकी
ह्याङ्की
स्टेफानोस्की
मोनार्की
सकिनसकी
नौटंकी
रामजानकी
हक्की
ह्विस्की
निस्की
टुकी
थकथकी
सकीनसकी
मास्की

भट्की
इराकी
अड्की
पानीट्याङ्की
सिद्दिकी
बान्की
हप्की
खोकी
फुलचोकी
शार्की
होकी
डुबुल्की
मुड्की
फर्की
धुकधुकी
नेहरु
खातिर
प्रहरी
मन्त्री
प्रधानमन्त्री
पोखरा
ढिलाई
खेले
उपस्थित
अनुपस्थित