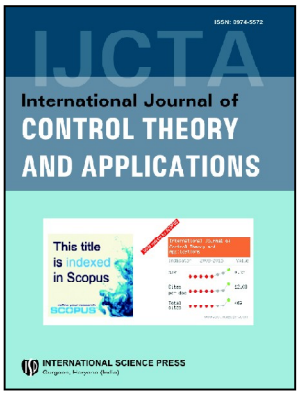

International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 10 • Number 10 • 2017

A Novel Approach to Telugu Stemming Using N-gram Process

N.V. Ganapathi Raju[1], Chinta Someswara Rao[2] and G. Meghana[3]

[1]Associate Professor, Dept. of CSE, GRIET, Hyderabad, INDIA.
[2]Assistant Professor, Dept. of CSE, SRKR Engineering College, Bhimavaram, INDIA.
[3]P.G. Scholar, GRIET, Hyderabad, INDIA
E-mails: [1]nvgraju@griet.ac.in, [2]chinta.someswararao@gmail.com, [3]meghanaraju92@gmail.com

Abstract: The ability of an Information Retrieval (IR) system to conflate the words allows reducing index and enhancing recall sometimes even without significant deterioration of precision. Conflation also conforms to user's intuition because users don't need to worry about the "proper" morphological form of words in a query. The problem of automated conflation implementation is known as "stemming" algorithm. One of the key solutions in conflation algorithms is if an automated stemming can be as much efficient (for IR purposes) as a manual processing. For this purpose in this paper, stemming methodology using N-gram is proposed which increases the efficiency of the Telugu IRS.

Keywords: Information Retrieval System; IRS; Stemming; N-gram; Conflation

1. INTRODUCTION

In linguistic morphology and information retrieval, stemming is the process of reducing inflected words to their word stem, base or root form—generally a written word form [1]. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. Algorithms for stemming have been studied in computer science since the 1960s.

Stemming is common form of language processing in most Information Retrieval (IR) systems [2]. Word stemming is an important feature supported by present day indexing and search systems. Idea is to improve recall by automatic handling of word endings by reducing the words to their word roots, at the time of indexing and searching. Stemming is usually done by removing any attached suffixes, and prefixes from index terms before the assignment of the term. Since the stem of a term represents a broader concept than the original term, the stemming process eventually increases the number of retrieved documents [3,4,5].

Stemming broadens our results to include both word roots and word derivations [6,7,8]. In natural language processing, *conflation* is the process of merging or lumping together non-identical words, which refer to the

same principal concept. It is commonly accepted that removal of word-endings (sometimes called *suffix stripping*) is a good idea; removal of prefixes can be useful in some subject domains (chemistry is an obvious example), but is not so widely practiced. The most obvious method for comparing the usefulness of Stemmers (Automatic programs that stem the word based on some algorithm) for the field of IR is by their impact on IR performance, using a testing system and a 'test collection' of documents, queries and relevance judgments. The process of stemming is important to the operation of classifiers and index builders/searchers because it makes the operations less dependant on particular forms of words and therefore reduces the potential size of vocabularies, which might otherwise have to contain all possible forms. It might be useful to think of stemming as the automatic definition of a group of synonyms for a particular word.

2. STATE OF ART

In this section different stemming algorithms are studied.

2.1. Lovins Stemmer

The Lovins stemmer [9,10] is a single pass, context sensitive, longest-match stemmer developed by Julie Beth Lovins of Massachusetts Institute of Technology in 1968. Lovins.

Stemmer maintains a list of most frequent suffixes, 250 in number, and it removes the longest suffix ensuring what the stem is at least 3 characters long. It is quite unreliable and frequently fails to produce the correct stem.

2.2. Porter Stemmer

The porter stemmer [11,12] is a conflation stemmer developed by Martin Porter at the University of Cambridge in 1980.Porter is designed for English language. Porter stemmer is based on the idea of suffixes of words are mostly made up of a combination of smaller and simpler suffixes. It has 5 steps applying rules with in each step. If a suffix rule matches a word, then the conditions attached to that rule are tested and the stem is obtained by removing the suffix. Porter stemmer is linear step stemmer. The Porter stemmer is readily available and widely used.

2.3. Dawson Stemmer

The Dawson stemmer [13,14] was developed by J.L .Dawson of the literary and linguistics computing centre at Cambridge University. It is based upon the Lovins stemmer, extending the suffix list to 1200 suffixes. It keeps the longest match and single pass nature of the lovins stemmer. It replaces recording rules which were found to be unreliable, using instead and extension of the partial matching procedure, also defined within the loving paper.

2.4. Krovertz Stemmer

The Krovertz stemmer [15,16] was developed by Bob krovertz, at the University of Massachusetts, in 1993. It is based on the morphology. Krovertz stemmer effectively and accurately removes inflectional suffixes and then checks a dictionary.

3. METHODOLOGY

In this paper, we proposed the N-gram based stemming method on Telugu language. The N-gram data structure is discussed in this section. The architecture of the N-gram based Telugu Stemmer for Telugu language has shown in Fig. 1.

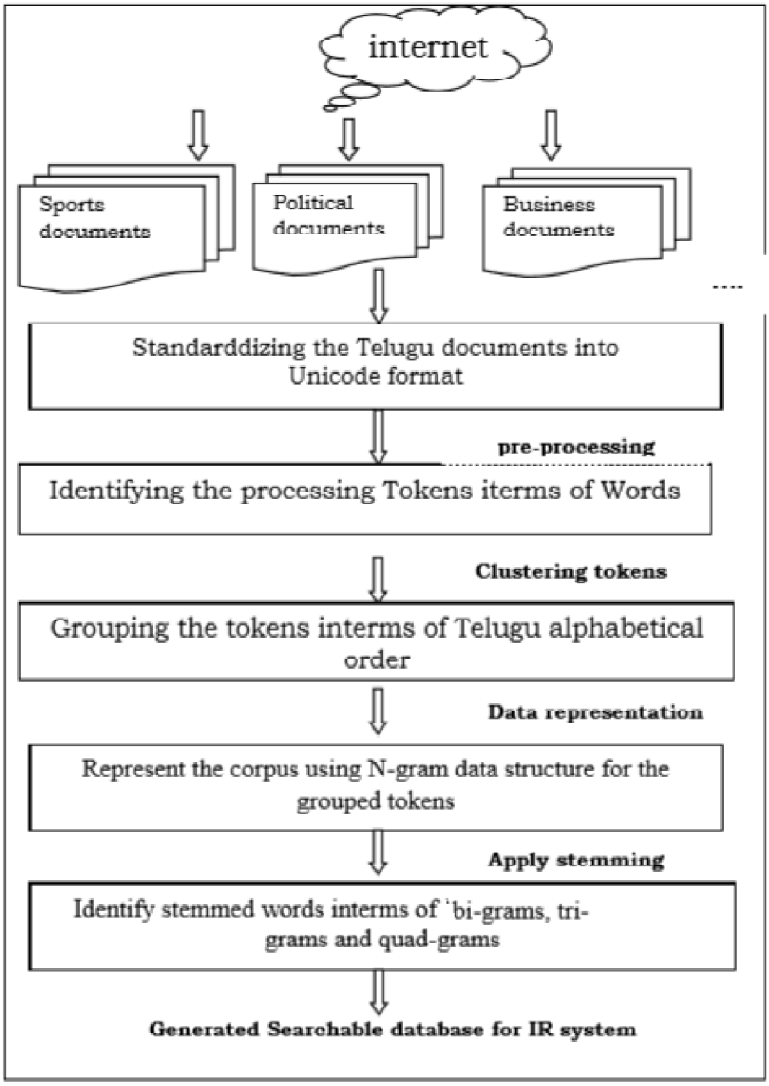

Figure 1: N gram based Stemmer for Telugu language

3.1. N-Gram Data Structures

The proposed algorithms for Information Retrieval are mainly concentrated are word based representation because there are several advantages for word-based systems over N gram based systems. First, the number of unique words is smaller than unique N grams (for n >3) in the same text corpus. As a result, the index for an N gram-based system will be much larger than that of a wosysrd-based system. Secondly, stemming techniques can be used in word based systems but not in N gram-based systems.

However, N grams [17, 18, 19] provide similar functionality by moving a window one character at a time, enclosing the stem alone at some time. Thirdly, N gram based systems don't explicitly remove stop words, though the importance of common N grams is reduced by standard statistical techniques such as cancroids subtraction or the use of TF/IDF weights. Additionally, N gram system doesn't make use of synonyms, though there is no reason why it couldn't be modified to do so. N grams are prone to be potential and outperform the traditional term frequency distribution where the token alphabet is relatively small and the representative set of all alphabets. A combination of naive Bayes classifier and statistical N gram model is reported in literature with surprisingly effective classifier performance. N gram language models and their applicability is widely studied

in statistical natural language modeling and speech recognition. N gram models are extremely simple and found to be effective in many applications. In text compression, character level N gram models are widely used Predication by Partial Matching (PPM). The PPM model is treated as a benchmark in text compression for decades. However, in speech recognition, research is focused on word level N gram models. These models are extensively used in text retrieval, particularly in Asian language text retrieval, such as Chinese and Japanese. N gram models are proposed for English information retrieval for phrase weighting in IR in text classification,

N gram models are investigated however, their model typically uses N grams as feature vector in a traditional feature selection process, and then explored by classifiers based on similarity measures among feature vectors. N gram indexing is reported to outperform raw word based indexing in many instances. In the case of morphologically complex languages like Indic scripts, N grams are assumed to be effective over raw words. It is likely that decompounding, stemming, or other normalization will result in better accuracy than the native use of raw words. With the help of N grams, statistical relationships between N grams in one language and those of a different language can be identified.

An n-gram is a subsequence of *n* items from a given sequence. The items in question can be phonemes, syllables, letters, words or base pairs according to the application. An *n*-gram of size 1 is referred to as a "unigram", size 2 is a "bigram" (or, less commonly, a "digram"), size 3 is a "trigram" and size 4 or more is simply called an"N-gram". There are two types in N-gram data structures one is overlapping N-grams where the second ones is non overlapping n-grams. In overlapping technique the characters at the end in the previous n-gram are repeated in the present and it continuous up to the end of the string where as in non overlapping technique the repetition of the characters which are already appeared in the previous n-gram are not allowed. The table 1 and 2 shows how to represent N-gram notations in English and Telugu languages.

Table 1
Representation of N-gram for English language when N=3

Example: n-grams for the word "good morning" with n=3 on English phrase
Overlapping n-grams: "goo", "ood", "od ", "d m", " mo", "mor", "orn", "rni", "nin", "ing"
Non Overlapping n-grams: "goo", "d m", "orn", "ing"

Table 2
Representation of N-gram for Telugu language when N=2

Example: n-grams for the word "తృతీయ శ్రేణి పట్టణాలకూ విస్తరణ కొత్త పుంతలు తొక్కిస్తాం" with n=2
Overlapping n-grams: "భార" "రత్" "తృతీ" "తీయ" "శ్రేణి" "పట్ట" "ట్టణా""ణాల" "లకూ" "విస్త" "స్తర" "రణ" "కొత్త" "పుంత" "తలు" "తొక్కి" "క్కిస్తాం"
Non Overlapping n-grams: "భార" "తృతీ" "శ్రేణి" "పట్ట" "ణాల" "విస్త" " రణ" "కొత్త" "పుంత" "తొక్కి"

4. RESULTS AND DISCUSSION

The system consists of Collection *of data from the web, Preprocessing and extraction of the words from the corpus, Clustering of the words, Construction of an N-gram data structures for each category, and Stemming of words using N-gram data structures* modules. These modules are discussed in this section.

4.1. Collection of Data from the Web

In this process collect the data from web in the following approach. Table 3 shows the process of documents collected from the internet.

Table 3
Collection of Telugu corpus

1. We collected the Telugu corpus (data) from the news papers which are available online in the Internet. There are some websites providing us the daily news papers in telugu script like www.uni.medhas.org.
2. From the internet the data for seven categories called business, editorial, literature, stories, sports and politics has been collected.
3. The data is converted in to Unicode format.

4.2. Preprocessing and Extraction of the Words from the Corpus

Pre-processing and extraction of words in the following approach. Table 4 shows the process of preprocessing and extraction of processing tokens from the Telugu documents.

Table 4
Shows the process of preprocessing and extraction of processing tokens

4. In this step, the files in the corpus are preprocessed where preprocessing is the process by which we will remove or ignore the characters which or other than in the Telugu language.The main task is to check the Telugu documents must be in Unicode range: 0C00–0C7F.
5. All the numbers, Roman characters and any unwanted letters except "space" will be removed.
6. This can be done in process described below.
   - 6.1. First, we read all the files of a category.
   - 6.2. Then, read the each file into a string variable.
   - 6.3 Next step is identifying each individual character.
   - 6.4 Remove the each character which is unwanted
   - 6.5 Final step is to write the string to a file.
7. From the preprocessed file we will extract the words using a space identifier which separates each word.

4.3. Clustering of the Words

In this step, the system will Group the words based on their first syllable. Table 5 shows the clustering or grouping of the processing tokens in Telugu languages alphabetical order.

Table 5
shows the clustering processing tokens based on alphabetical order

1. For each word the first syllable is checked and if a group named with that syllable is doesn't exist it will create a new group with the name of that first syllable and it will push that corresponding word in that group.
2. If a group with the name of that first syllable already exist push that word into that group.
3. For each group a new file is created
4. Ex. For a word -/.A after syllabification it becomes as - / .A
5. The first syllable is - so we will create a group called group-.txt

4.4. Construction of N-gram Data Structure For Each Group

In this step, we will construct an N-gram data structure for each group. Table 6 shows construction of N-gram data structure for each group.

Table 6
Shows construction of N-gram data structure for each group.

1. N-gram data structure is continuous division of string into a fixed length N. N-grams may be of type overlapping and non-overlapping her we use the technique of non-overlapping.
2. Based on the length of N we name the n-grams as Unigram, Bigram, trigram and Quad gram for N=1,2,3.4 respectively
3. To store all these us use sets in python which will be useful to remove the redundant N-grams automatically.
4. Each set is named as Bidict, Tridict and Quadict for Bigrams, Trigrams and Quad grams Respectively
5. These sets are used further in the stemming process

4.5. Stemming of Words Using N-gram Data Structures

It is the final step using those three sets, we develop another three dictionaries having key as stem (bigram/ trigram/quadgram) and value as words for those corresponding N-grams. Table 7 shows the stemming process for the words using N-gram data structure.

Table 7
Shows the stemming process for the words using N-gram data structure

1. Here we will give priority order as higher priority to quad gram, priority levels towards trigram and bigram having less priority.
2. Starting with quad gram if a key in quad gram having two or more words as value then we will consider the key as a stem and push into a stemmed dictionary
3. The above step is repeated for tridict and bidict.
4. Now stemmed dictionary containing some key value pairs by giving higher priority to quadgram we will remove if any other values in the dictionary contains same words which under come in the values of quad-gram.
5. If a word is exactly of length 2 or 3 or 4 and doesn't have any morphological variants then that word itself is considered as root.

5. ALGORITHMS USED IN IMPLEMENTATION

The proposed stemming process has two algorithm named as algorithm for clustering of the words and algorithm for stemming using N-gram data structures. The tables 8 and 9 shows the algorithms for clustering of the words and stemming using N-gram data structures

Algorithm 1: Algorithm for clustering of the words

1. Read all the files from a category
2. Preprocess all the files
3. Divide the files into words
4. For each word
5. Begin
   5.1. Do syllabification
6 For each word
7 Begin
   7.1 Check first syllable
      7.1.1 If group exist
      7.1.1.1. Add word to that group

7.1.2 Else
7.1.3 Begin
7.1.3.1 Create group
7.1.3.2 Add word to that group
7.1.4 End
8 End
9 End

The tables 8 algorithm for clustering of the words

Algorithm 2: Algorithm for stemming using N-gram data structures

1 For each group
2 Begin
2.1 Extract words
2.2 For n=1 to 3
2.3 Begin
2.3.1. For each word
2.3.2. Begin
2.3.2.1. Split the word into length n
2.3.2.2. And add to corresponding dicts
2.3.3. End
2.4 End
3 For each dictionary
4 Begin
4.1. Set priority for dictionary
4.2. Check for uniqueness of values by comparing with other two dictionaries
4.3. push the unique values along with keys into new dictionaries
5 End
6 Look in new dictionary
7 Begin
7.1. If a Key with two or more words in value
7.2. Begin
7.2.1. Consider as final stem
7.3. End
8 End
9 End

The table 9 shows algorithm for stemming using N-gram data structures

6. SIMULATIONS

In this paper a tools were developed for the preprocessing of the Telugu documents and for stemming of Telugu words using N-gram is developed which are shown in Fig 2 and Fig 3. The example result of a word అధికార has been given in the Fig 4.

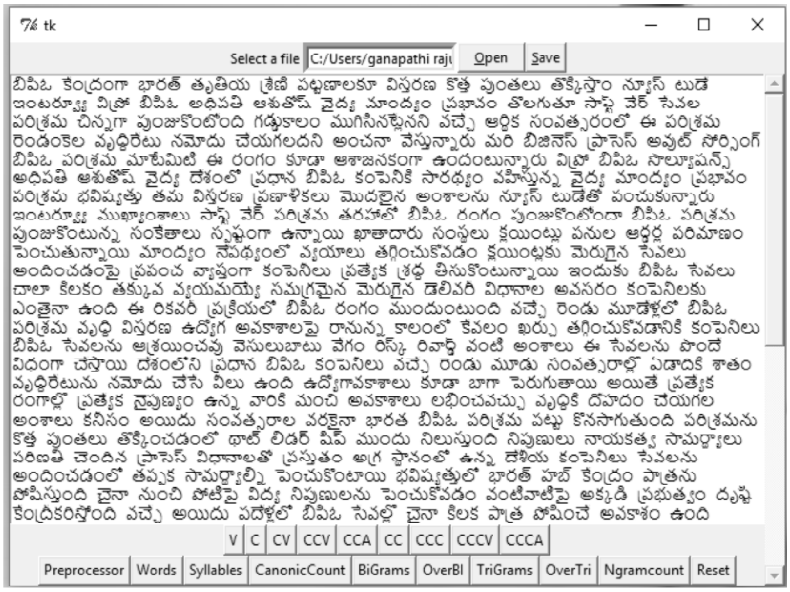

Figure 2: Tool for Preprocessing and Tokenization for Telugu documents

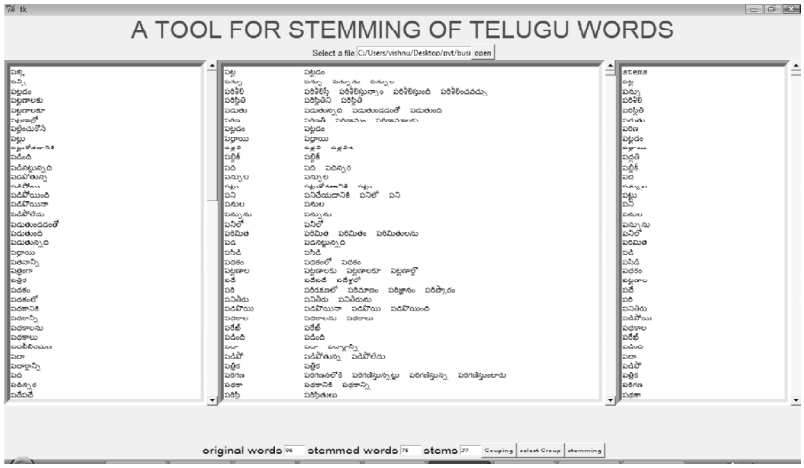

Figure 3: Stemming tool for Telugu tokenized words

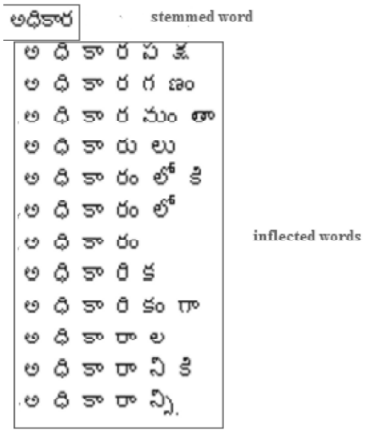

Figure 4: Stemmed word and Inflected words for a అధికార

For evaluating the efficiency or performance of our N-gram Stemmer, totally 350 Telugu documents are considered for the experimentation which includes seven kinds of categories of Telugu documents. Each category consists of 50 documents each. The categories are business, editorial, literature, rivers, science, and stories.

Table 10
Results after applying N-gram based Telugu Stemmer on Telugu documents

Topic/group	*Size in KB*	*Total number unique words(before Stemming)*	*Total number unique words (after Stemming)*	*Number of words per conflation*
business	67 kb	2777	1305	2.12
editorial	245 kb	9159	4067	2.25
literature	833 kb	4750	2055	2.31
science	296 kb	2861	323	2.16
stories	560 kb	4529	2142	2.11
rivers	332 kb	9265	4240	2.18

Table 10 clearly shows that our N-gram based Telugu stemmer on average conflation ratio is 2.18 words on all six categories. This reduces the size of the searchable data structure. Fig 5 has shown that the total number of words extracted topic wise before and after stemming.

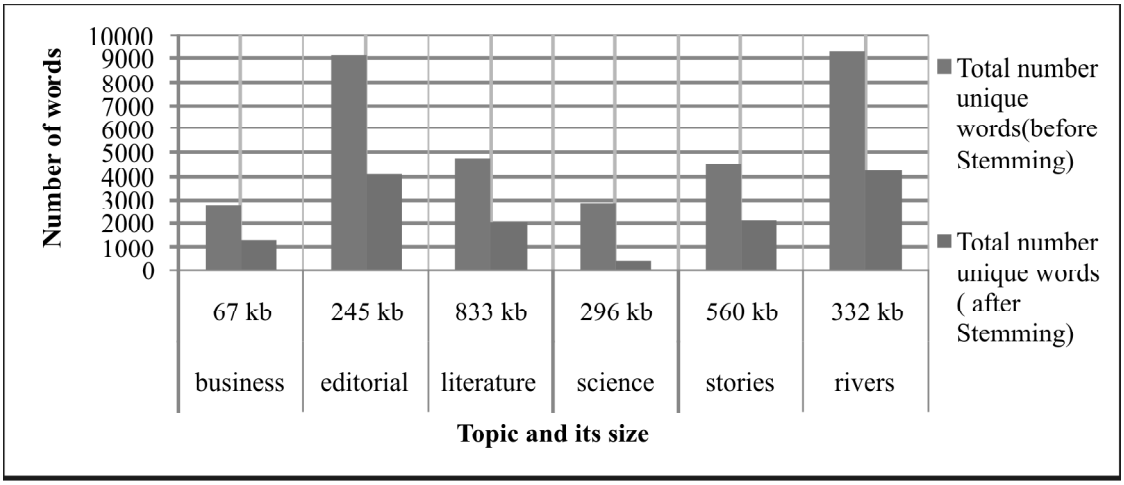

Figure 5: Total number of unique words extracted before and after stemming

7. CONLUSIONS

This paper presents an unsupervised approach for the development of a stemmer (For the case of TELUGU language). Especially, during last few years, a wide range of information in Indian regional languages has been made available on web in the form of e-data. But the access to these data repositories is very low because the efficient search engines/retrieval systems supporting these languages are very limited. Hence automatic information processing and retrieval is become an urgent requirement. For this purpose in this paper covers major part of search engine implementation like stemming. The results show that the proposed paradigm out performs after applying N-gram. Proposed stemmer is unsupervised and language independent. Corpus is used to derive set of powerful suffixes and it does not need any linguistic knowledge. Hence, this approach can be used for developing

stemmers for other languages that are morphologically rich. The future work will carry by storing the stemmed documents in Inverted File Structure and make it easy for retrieving relevant documents based on the query using page ranking algorithm.

REFERENCES

[1] Naji N, Allan J. On Cross-Script Information Retrieval. InEuropean Conference on Information Retrieval 2016 Mar 20 (pp. 796-802). Springer International Publishing.

[2] Makhija SD. A study of different stemmer for Sindhi language based on Devanagari script. InComputing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on 2016 Oct 31 (pp. 2326-2329). IEEE.

[3] Saharia, Navanath, Utpal Sharma, and Jugal Kalita. "Stemming resource-poor Indian languages." ACM Transactions on Asian Language Information Processing (TALIP) 13.3 (2014): 14.

[4] Zhai C, Massung S. Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining. Morgan & Claypool; 2016 Jun 30.

[5] Büttcher S, Clarke CL, Cormack GV. Information retrieval: Implementing and evaluating search engines. Mit Press; 2016 Feb 12.

[6] Flores FN, Moreira VP. Assessing the impact of Stemming Accuracy on Information Retrieval–A multilingual perspective. Information Processing & Management. 2016 Apr 18.

[7] Shrestha I, Dhakal SS. A new stemmer for Nepali language. InAdvances in Computing, Communication, & Automation (ICACCA)(Fall), International Conference on 2016 Nov 21 (pp. 1-5). IEEE.

[8] Thangarasu, M., and R. Manavalan. "A Literature Review: Stemming Algorithms for Indian Languages." arXiv preprint arXiv: 1308.5423 (2013).

[9] Julie Beth Lovins (1968).Development of a stemming algorithm. Mechanical Translation and Computational Linguistics 11:22—31. 1977

[10] Ismailov A, Jalil MA, Abdullah Z, Rahim NA. A comparative study of stemming algorithms for use with the Uzbek language. InComputer and Information Sciences (ICCOINS), 2016 3rd International Conference on 2016 Aug 15 (pp. 7-12). IEEE.

[11] Porter, M. (1980). "An algorithm for suffix stripping program", Vol. 14, pp. 130—137.

[12] Rajalingam M, Raman V, Sumari P. Implementation of vocabulary-based classification for spam filtering. InComputing Technologies and Intelligent Data Engineering (ICCTIDE), International Conference on 2016 Jan 7 (pp. 1-6). IEEE.

[13] DAWSON, J. 1974. "Suffix Removal and Word Conflation." ALLC Bulletin, Michelmas, 33-46.

[14] Singh J, Gupta V. A systematic review of text stemming techniques. Artificial Intelligence Review. 2016:1-61.

[15] Krovetz R. Viewing morphology as an inference process. InProceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval 1993 Jul 1 (pp. 191-202). ACM.

[16] Lande P, Dalal V. Analyzing Social Media Data to Explore Students' Academic Experiences. International Journal of Computer Applications. 2016 Feb; 135(4):13-6.

[17] Korpal R, Bose S. A Framework for Detecting External Plagiarism from Monolingual Documents: Use of Shallow NLP and N-gram Frequency Comparison. InProceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies 2016 Mar 4 (p. 61). ACM.

[18] Singh J, Gupta V. Text Stemming: Approaches, Applications, and Challenges. ACM Computing Surveys (CSUR). 2016 Sep 16; 49(3):45.

[19] Ehsan N, Tompa FW, Shakery A. Using a dictionary and n-gram alignment to improve fine-grained cross-language plagiarism detection. InProceedings of the 2016 ACM Symposium on Document Engineering 2016 Sep 13 (pp. 59-68). ACM.