An Application of MDL Principle for Indian Resource Poor Language

Miral Patel
Charotar University of Science and Technology, Gujarat, India
and
Apurva Shah
Sayajirao Gayakwad University, Gujarat, India

Stemmer is very important and required module for any morphological system. Stemming process is language dependent, which separates stem and suffix from a given word. Even after notable growth, specifically work at morphological level for Indian resource poor languages like Sanskrit, Assamese, Bengali, Bishnupriya, Manipuri, Bodo etc. are less attended. Standard resources (corpus, data set) for experiment are very scarce for such languages. Many famous unsupervised approaches are tested for European languages only. It is the requirement to see how well famous approach works for other inflective and resource poor languages. In this study, Minimum Description Length principle (MDL) is applied to Sanskrit (resource poor and inflective) language. Initially, all corpus lexicon are split in to substring, which is followed by calculating frequency and length of each sub string. A higher probability split is considered as best split for stem and suffix. Next, multiple iteration is taken until result improved. With 72 % result MDL works well for Indian language. MDL principle is extended to improve performance of Sanskrit stemmer by adding rule based approach. MDL based hybrid approach improves result by 17 %. As no direct Sanskrit stemmer or evaluation is available to compare, therefore, we compare our work with Lovin, Porter and Paice stemmers. Word stemmed factor is highest compared which to all three stemmer. Our results are also comparable to Gujarati and Punjabi language stemmer. Stemmer strength is more as it reduces under stemming errors.

Keywords: Stemming resource poor languages, application of MDL, suffix striping, suffix removal, unsupervised stemming and statistical approach for stemming..

1. INTRODUCTION

Normalization of different morphological variants of the word is very significant and vital preprocessing technique in NLP. Which is very important initial preprocessing task in morphological analysis. Various applications like transliteration, information retrieval (IR), machine translation (MT), spell Checker, question answering system (QAS), information extraction (IE) and language modeling have a requirement to get the root word. Stemming requires getting correct root from variant forms of the word. Contemporaneous stemming algorithms belong to rule based, statistical and hybrid (see section 2). The majority of work is done in European languages for all approaches, gradually work started for other languages. A large pool of languages, almost 7000 languages are available worldwide is provided in Hammarström and Borin [2011] . Number of inflective natural languages are available e.g. Indo-European, Uralic and Semitic etc Inflective languages are having tendency to map different morphological forHms to same topic given in Smirnov [2008]. Yet, many highly inflective languages are unconsidered for stemming. It is to be discovered, whether the present rule based approaches and other statistical approaches are applicable to highly inflective and resource poor languages or not.

Sanskrit is an ancient classical Indian language and considered as the mother of all Indo-European languages. It is also one of the official languages out of 22 declared languages in India. Old Sanskrit defined by PANINI is known as Vedic Sanskrit. Later, it took a form of classical Sanskrit. The nature of language is very inflective with rich morphology and hence, has

more suffix and prefix in the language system. Sanskrit consists of many linguistic challenges, where it requires special care for morphological processing. There are orthographic, prosodic and inflectional complexities in Sanskrit, which is not encountered in western European languages. Sanskrit is prosperous by the way of having inflectional and derivational morphology. Which permits association that is exposed in European languages where understanding of morphology is required. Being a word order free language, to fix the meaning of the word by its position in sentence is not always true for Sanskrit. Sanskrit is very flawless and ample than the Greek and Latin structure.

In this research paper, the problem of stemming is considered for resource poor language (In particular Sanskrit). Resource poor language having less or no standard recorded data set, corpus, and document repository available for research work. Also, lack of tools or any benchmarked system on which base experiment can be implemented. Initially, our aim was unsupervised approach. Based on successive application of different heuristic, requirement of the system has become clear. System needed kind of good reference from earlier result/signature, which can help in stem and suffix identification process. Such a reference can be provided by adding rule based approach (hand crafted suffix list for 1K words was prepared). Which makes developed system a hybrid stemmer.

Motivation to work for resource poor language was very high after completing literature survey and requirement to see applicability of existing approach was also need of time. Moreover, in modern world of digitization it is indeed constraint to make available all Sanskrit resource from old useful literature to public. Later, it can also be transferred to different languages and its morphological system. Our contribution of developed stemmer can be used in any morphological system. Morphological system requires different phases like (1) Tokenization, (2) Stop word removal (3) Stemming (4) Phrase recognition (5) Term weighing. Stemming and stop word removal is language dependent module which is the requirement to develop such module for any morphological system.

2. BACKGROUND AND RELATED WORK

Rule based approach requires language expertise for forming the set of rules which is very time consuming process. Statistical approach does not need language knowledge as it employs a statistical information from a large corpus with different methods like HMM, MEM, N-gram, Co-occurrence, YASS, GRASS and MDL. However, these approaches require a large corpus collection that covers most of morphological variants; otherwise, accuracy is highly compromised. Statistical stemmer cannot handle many exceptional cases; (mouse: mice, put-put, and foot: feet etc). So, in this case rule based stemmer outperforms statistical approach.

Lovins [1968], First ever developed rule based stemmer is very easy, fast and employs iterating and longest string matching approach for English Language. Porter [1980] is the best stemming algorithm in its time, comprised of a set of rules employed in English language. Snowball Porter [2001] is a language for stemming algorithms which allows development of stemmer methods for different languages. Paice [2006] has compared algorithm with Porters stemmer and found that his approach has a tendency to over-stem (being a heavy algorithm is his terminology).

Number of unsupervised approaches are adopted for stemming, but in this paper, only famous and benchmarking techniques are considered. Mayfield and McNamee [2003] technique suffers from performance penalty, N gram statistical approach demonstrates a language neutral approach. Next, in series authors report improved runtime penalty with significant performance improvement for N gram. Proposed method is working well with a language where no morphological tool kit is available in McNamee and Mayfield [2007].

In Majumder et al. [2008], implemented (YASS) a suffix striper for Bengali, French and German languages . For classification, agglomerative clustering algorithm was used which improves stemming with compare to Porters and Lovins algorithm. In addition, improves stemming of French and Bengali compared to no stemming approach. Paik et al. [2011] Known as GRASS, a graph based stemmer which has specific focus on the collections of distinct lexicon. An Experiment was carried out for seven languages. GRASS outperforms YASS in a manner more number of languages are considered and results are also more accurate. **?**] shows Co-occurrence of words in documents and graph based statistical stemmer famous work reported in literature . Authors evaluate stemmer on European (Czech, Hungarian, Bulgarian, and English) and Asian languages (Marathi, Bengali).This stemmer outperforms YASS and window based stemmer, it has also reported comparable performance to other stemmer. A novel corpus based method

In Bhamidipati and Pal [2007] models the given words, as generated from a multinomial distribution over the topics available in the corpus. This procedure includes sequential hypothesis testing, which enables grouping together distributional similar words. Superiority is measured with respect to four existing stemmer.Caumanns [1999] reports stemmers for morphological complex languages like German and Dutch . This algorithm deals with character and character sequence in its first part, which takes linguistic rules and statistical heuristics into consideration. While second step works for application of context free suffix-stripping algorithm, which is scalable with more rule set and updated heuristic.

In Goldsmith [2001] a standard technique of frequency and length based algorithm Minimum Description Length is developed. MDL is applied to European languages and able to handle 5K to 500K word set. List of iterative heuristic is applied, which generates the morphological grammar. Next, MDL principle is applied to check updated heuristic works according to intuition. Results are very much progressive for this method. In Hammarström and Borin [2011] very critical and exhaustive study of stemming methods is provided in survey article . Author explores very refined and outstanding information of developed stemmers so far. Detailed study is carried out for various level and approaches for existing stemmer implementation. Authors have also provided future direction for work and have recorded fact that, scarce and resource poor languages are not attended which is requirement to justify applicability of any technique on one particular language. Saharia et al. [2013] implemented HMM based stemmer for Assamese; resource poor and highly inflective language. Author Saharia also supports the conclusion of Hammarstrm; about very little work is reported for Indic resource poor language.

We provide brief introduction in section 1. Section 2 provides glimpse of Sanskrit morphology. With section 3 we move ahead with an algorithm steps description and implementation of the proposed method. Section 4 offers detailed performance evaluation and Section 5 is dedicated for conclusion. Section 6 presents future work.

2.1 Sanskrit and Morphology

The entire literature survey reveals that contribution for Sanskrit language is very rare. In Brigs [1985] presented a very comprehensive report on how Sanskrit can take place for most suitable language for machine translation and how it may work for unambiguous knowledge representation for NLP. PANINI; the author of (Form of Sanskrit grammar) formulated 3,949rules for Sanskrit.Sanskrit grammar consists 7 cases and 3 numbers; therefore in Sanskrit each word has at least 21 forms of the word. Including vocative case, 24 forms per word is created where each word form has a different meaning. Sanskrit does not have word order as an issue like English, as it is word order free language. Verb in Sanskrit is considered as dhaaturuupa. Each verb consists of 10 tenses and 3 person. Noteworthy and quantifiable material about Sanskrit is presented like inflectional morphology, unique features of grammar and challenges of Sanskrit grammar.

In Goyal et al. [2012] concluded by emphasizing that, statistical approach is required for Sanskrit computation. Insufficiency of contemporary computational methods is gaudily apparent in the analysis of Sanskrit. However, Sanskrit morphological analyzer is available in Jha et al. [2009]. Sanskrit Analysis System (SAS) is presented in Bhadra et al. [2009] . Formal structure of Sanskrit text is very nicely presented in Huet [2009]. Author had also proposed interesting future work for Sanskrit processing system and concluded that, optimization problem can be solved with statistical approach. Although, we did not find clear mentioning of result or covered data set used in implementation. We have been motivated from the Minimum Description Length (MDL) principle available in Goldsmith [2001]. Basically MDL principle works best for compression of data by learning the regularity in available data set. Later, we have extended the algorithm by providing hand crafted suffix to improve accuracy.

3. ALGORITHM

Basic terminology: W= word, w =weight, S= stem, Su= Suffix, i= Length of stem, L= Length of word, (L-i) = Length of Suffix, f= Frequency, D = Database, Sig= Signature

Algorithm 1 Frequency and length based computation

Input: Corpus which contains list of Sanskrit lexicons
Output: Stem of lexicon

1. For each Word W do
2 Split W for all possible segmentation according to maximum word length;
3.. For each segmentation of W do
4. Find f(S), f(Su),
5. `F(i)=`$i * log$`(freq_of_stem)+(L-i)*log(freq_of_suffix))` (1)
6. Until segmentation ends for W
7. Find optimal split
8. Update S and Su list
9. Verify spurious signature based on handcrafted suffix list
10. Remove false signature
11. Update Sig list in D
12. Until entire corpus covered

3.1 Experiment Detaill

For experimental setup, Resource for Sanskrit documents received from Jawaharlal Nehru University (Delhi). Almost 1500 Sanskrit words were extracted randomly, few preprocessing steps are applied to make corpus suitable for experimental setup. Transliteration system is used to present corpus in Devanagari script. Duplicate words are removed therefore, only distinct words are available in corpus. Corpus is used to learn statistical knowledge. Developed system can take input for Sanskrit word and generates stem and suffix as output of word.

MDL Principle: As a resource poor and inflective language, Sanskrit requires to be attended by research community. MDL (see section 2) principle works on three main elements in the system, these are stem, suffix and signature. It also provides benefits of unsupervised technique and saves memory by storing stem and suffixes in signature form; which is very useful in inflective languages. Frequency and length is calculated for each segmentation. Initial heuristic is taken as All split and later successively iterates over different heuristic based on result.

Signature: Signature is the list of stem and suffix where mapping of one stem to different suffix is retrieved. Such different signature sets can be formed based on number of experiments, which is really advantageous to get the reference for unknown word provided to the system for stemming. In point of fact, this approach saves memory and computational time both. Goldsmith [2001] provides very detailed description about Minimum Description Length Principle (MDL) .

As mentioned earlier, Sanskrit is inflective and morphologically complex language. Therefore, it is not possible to cover entire length of the word. Therefore, majority case of the language, which is 6 character is considered. However, to see the performance of other length of word, we have taken few runs and analyzed result in later section.For experiment, we split word in to all possible segmentation by taking heuristic all split. Next, length and frequency of all segmentations is calculated. Eq. 1 as given in algorithm a is used to find the probability of the stem and suffix. To able to understand how to find, S, Su and F (i), Table I represents one of the examples of word modeling. Generated stems and suffixes are stored in database for signature generation and further reference for future runs. Next, optimal spilt is determined based on probability calculated as shown in figure 1, which shows graph for two sample words.

ID	Suffixlen	Stem	FreS	Suffixlen	Suffix	FreSuffix	Value
1	1	b	4	5	Alako	1	0.60205999
1	2	bA	4	4	lako	1	1.20411998
1	3	bAl	4	3	ako	1	1.88617997
1	4	bAla	4	2	k	1	2.40823997
1	5	bAlak	4	1	o	2	3.31132995
1	6	bAlako	4	0	-	1	0.00000000

Table I: Example of word modeled using frequency and length based approach

ID	Stem	Suffix	Function value	Observatrion
14	lawA	yAH	4.74269372	AyAH is actual suffix
22	muKA	Ya	2.89431606	Correct suffix
38	lawA	su	2.40823997	Correct suffix
62	siwA	yAH	5.13033377	Correct suffix
68	kan	yA	2.98766626	A is actual suffix
73	niSA	ByaH	4.31672498	Correct suffix
81	ka	lamAw	2.15836249	Correct suffix
82	aXyAoikA	yAH	7.15093368	Correct suffix

Table II: Example of some resultant stem-suffix pair after implementation

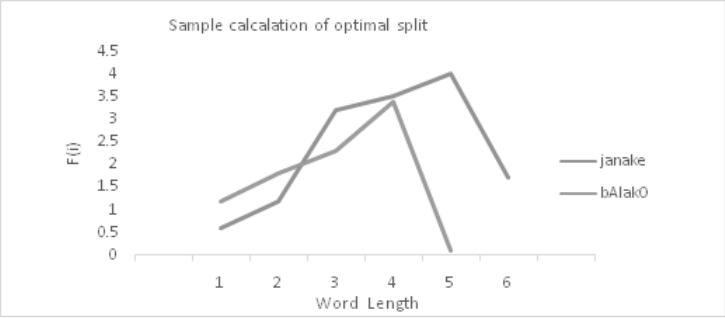

Figure 1: Selection of optimal split

Word length on X axis and probability retrieved using Eq. 1 is taken on Y Axis. Observation of graph indicates, value of words for different slices is increasing. Eventually after reaching top it falls down, which is considered as an optimal split. For example, "bAlakO"(boys) =¿ "bAlak" (Stem)+ "O"(Suffix) and for word "janake"(father) =¿"janak"(stem) + "e"(suffix) respectively. Obtained stems and suffixes are accurate hence, optimal split correct. Split largely depends on frequency distribution of the stems and suffixes of the word, language and corpus. For each such word in corpus we get stem and suffix based on optimal split and update frequency of stem(S) and suffix (Su).

To update stem and suffix, first we consider reference from system, if it is repeated duplicate entry is not allowed but, if new stem or suffix is found then it is stored with new ID in data base. So, each stem and suffix have its own ID. After each spilt, if heuristic value is 0 that particular stem and suffix is not considered for a member to generate signature, for accuracy reason. After, successive iteration Signature set is retrieved. For e.g. in signature "Sanskriti" (Heritage), "Smuriti" (Memory),"Prakruti" (Nature), "mati" (Intellectual) word maps it to suffix "ti". Signature provides common suffix for different word or for second case, different suffixes for common stem. Signature element provides the benefit of compact representation of corpus and hence gives advantage of less memory required for storage and processing of an algorithm. Table III displays some resultant signature set derived from implementation. Different suffixes associated with stem ID 2 are Suffix ID 11, 14, 19, 5, 22.

No linguistic knowledge was considered as experiment was carried for unsupervised approach. By observing signature in database, some unwanted signatures have been formed during computation (Discussed in section 4). Signatures with one stem and one suffix are removed because value of logarithm is always 0. Only polished signatures are stored for further computation. Add on contribution is, to provide handcrafted suffix to test and improve the performance of stemmer. Once signature is generated, verification is done with handcrafted suffix list. For invalid suffix, more iterative heuristic is applied for next possible suffix with other value of function. Sanskrit word takes one letter prefix while noun represented in past form. Added second step also deals with such words to remove its prefix with suffix. Observation and conclusions for experiments are reported in next section.

Signature ID	**Stem ID**	**Suffix ID**
1	1	10
2	2	11
	2	14
	2	19
	2	5
	2	22
2	3	7
	3	21
	3	18
	3	15
	3	19
	3	10
4	4	15
	4	3
	4	21

Table III: Example of signature generated

4. PERFORMANCE EVALUATION

By analyzing corpus and generated signature set, incorrect stem and suffix pairs are generated. Hence, it has formed erroneous signature. Few such results are recorded in Table III. Based on wrong signature set, any further experiment would lead to propagation of error for future runs. Although AyAH is correct suffix for word lawAyAH (see TABLE II) but, implementation assigns higher value 4.7 to yAH as suffix which is not correct. Another inspection recorded is, word kanyA where correct suffix is A but, Resultant suffix is yA highest value 2.8. Imparted rule based approach with handcrafted suffix was aim to remove such kind of errors. Provided handcrafted suffix helps to provide reference from suffix list and hence wrong suffix and stem possibility was reduced. After applying rule based approach more 17 % accuracy was gained. This is mark able for resource poor languages like Sanskrit, where no prior unsupervised or hybrid stemmer was reported. Although hybrid approach provided mark able gain in accuracy, our system reported over stemming error more compared to under stemming. So, that way strength of stemmer is aggressive.

Asymptotic complexity for the algorithm to run t (n) is O (n). Execution time for initial run was reasonable. While word segmentation increases computational time is also increasing. Very first run of experiment was pure unsupervised approach where reported accuracy of 72 %. Out of 1500 words, correctly stemmed words are 1090. Table V contributes towards performance evaluation and analysis based on different criteria. Strength of stemmer can be measured by index compression factor (ICF). Stemmer strength is more for pure unsupervised approach while hybrid approach gives less aggressive. Therefore, improves accuracy by reducing some over stemming errors. Superiority of hybrid approach is confirmed by higher word stemmed factor (WSF) and correct stemmed word (CSW).Correctly stemmed word factor is higher for unsupervised technique but difference with hybrid approach is extremely comparable. Average word conflaction factor (AWCF) does not vary much while WSF and CSWF results are notable. Over all accuracy is higher for hybrid approach with WSF 96.66%. Though, CSWF value is higher for unsupervised approach; because it has stemmed some of the root word; detailed parameter is listed in Table IV. Table V represents comaprison of developed sanskrit stemmers with other popular stemmer.

System was also tested for different length of word more than six character. As shown in Figure. 2, graph for word length vs. accuracy. Words of length 12 and 15 are presenting good result due to very less number of words are available for such length in corpus and hence it increases result. Otherwise, results are very poor for greater length of word.

System Configuration: Language: JAVA 1.6 and My SQL database. Platform: 32 bit Windows 7 Professional, Processor: Intel core I3 CPU, Memory: 4GB.

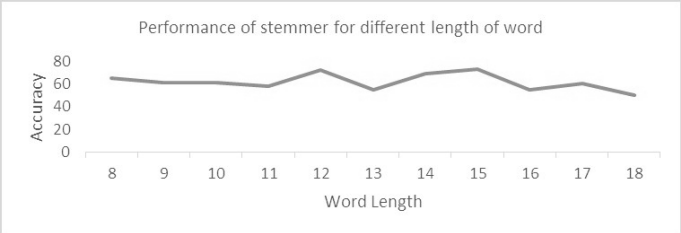

Figure2: Performance of varying length of word

Terminology for strength and accuracy parameter:

a) **Index Compression Factor:** ICF = n-s/s; Where n = no of words and s is number of stems in corpus. As ICF value is increasing stemmer strength is also increased.

Parameter	Unsupervised	Hybrid
Total number of word	1500	1500
Number of distinct word before stemming (N)	1500	1500
Index Compression Factor (ICF)	37.61%	15.74 %
Number of words Stemmed(WS)	1180	1450
Words Stemmed Factor (WSF)	78.66%	96.66%
Correctly Stemmed words (CSW)	1090	1296
Incorrectly stemmed words (ISW)	410	204
Correctly Stemmed words Factor (CSWF)	92.37%	89.37 %
Correct Words not stemmed (CW)	98	81
Average words conflaction factor AWCF	-5.50%	-5.63%

Table IV: Evaluation of developed Sanskrit stemmer

Parameter	Unsupervised	Hybrid	Lovin	Porter1	Porter2	Paice
Number of Words	1500	1500	1858	1858	1858	1858
ICF	37.61	15.74	48.29	42.4	42.4	48.83
AWCF	-5.50	-5.63	-24.81	-8.52	8.6	19.26
WSF	78.66	96.66	73.35	67.17	66.58	70.99
CSWF	92.37	89.37	27.80	31.97	34.76	28.73

Table V: Evaluation of developed Sanskrit stemmer

Author	Language	Dataset	Technique	Accuracy
Saharia et al. [2013]	Assamese	50.000	Hybrid	92%
Saharia [2010]	Assamese	-	Unsupervised	85%
Nehar et al. [2012]	Arabic	-	Unsupervised	-
Sheth and Patel [2012]	Gujarati	-	Rule based	-
Ameta et al. [2011]	Gujarati	3000	Rulebased	91.5%
Suba et al. [2011]	Gujarati	EMILLE	Hybrid	90%, 67.7%
Dolamic and Savoy [2009]	Czech	78 MB	Unsupervised	-
Kumar and Rana [2010]	Punjabi	52,000	Unsupervised	81.27%
Ramanathan and Rao [2003]	Hindi	35997	Rulebased	88%
Amaresh Kumar Pandey [2008]	Hindi	35997	Unsupervised	90%

Table VI: Some of the reported stemmer for inflective languages with accuracy

b) **Average Word Conflation Factor:** AWCF represents accuracy of stemmer. Higher the percentage of AWCF, will lead higher accuracy of stemmer. It shows the out of multiple variants of word how many variants are actually stemmed to root.

c) **Word Stemmed Factor:** WSF represents total words stemmed (correct or incorrect) out of provided words in corpus. If value of WSF is higher, stemmer strength is considered to be more.

d) **Correctly Stemmed Word Factor:** CSWF indicates percentage of words stemmed correctly out of stemmed words. Higher the value of CSWF accuracy is also higher. More detailed explanation available in ¡]Sirsat2013 , ¡]Frakes2003.

In literature survey, for Sanskrit language, no specific stemmer for unsupervised or hybrid approach is reported. Therefore, it is not possible to compare results with same language. Comparison of some well-known rule based stemmer with Unsupervised (U) and Hybrid (U+R) stemmer is provided in Table V. It clearly shows that, hybrid stemmer with WSF 96.6% is leading in all reported stemmer. Lovin and Husk stemmer is highly aggressive with 48%, but unsupervised approach is less aggressive so. Hybrid approach reduced ICF with 15.54%, which shows added rule based approach gives light stemmer with reducing over stemming errors. Unsupervised

Input Word	Overstemming	RootWord as input	English Transformation
तदा	त+ दा	तदा	Therefore
पद	प+ द	पद	Position
अत्	अ+ त्	अत्	Here
भगवान्	भग+ अवान्	भगवान्	GOD
भआगीरथी	भ+ आगीरथी	भआगीरथी	Hard working
दक्षिण्प्रदेशे	दक्षिण्+ अप्रदेशे	दक्षिण्+ अ+ प्रदेशे	West region
पाटलीपुत्रअपर्यन्त्	पाटलीपुत्र+अपर्यन्त्	पाटलीपुत्र+अ+ पर्यन्त्	Nextgeneration of king
उत्तरदिशायां	उत्त+ रदिशायां	उत्तर+ दिशायां	In east direction
सव्प्रथमं	सर्व्+ प्रथमं	सर्व्+ अ+ प्रथमं	Very first
सूर्यास्तसमये	सूर्यस्त+ समये	सूर्य+ अस्त+ समय+ +अ	At the time of sunset

Figure4: Few stemming errors

approach gives highest WSF among all stemmer in Table V. AWCF is lowest for both approaches.

Furthermore, Table VI provides over all comparison with other resource poor and inflective language. Higher accuracy is gained compared to Gujarati language on unsupervised approach. All over accuracy is higher compared to Punjabi and comparable to Assamese language.

5. CONCLUSION

Primary goal to develop preliminary unsupervised stemmer for resource poor and inflective language Sanskrit is achieved with accuracy of 72%. Based on experiment results and analysis, intuition gain was; with some support of suffix reference wrong suffix generation can be controlled up to major extent and that made us to extend the scope of work by adding hand crafted suffix. Here, stemmer becomes hybrid by adding rule based approach to unsupervised approach. Result proved that analysis was done on proper direction as stemmer has increased accuracy to 86.4% after adding the hand crafted suffix. Unsupervised stemmer removes only suffix but, Hybrid stemmer also removes prefix. Strength of stemmer for pure unsupervised stemmer is recorded high and hence, it is aggressive compared to hybrid stemmer. However, hybrid stemmer has reported high WSF compared to all stemmers (Word stemmed factor reflects how accurately words stemmed) and unsupervised stemmer scores second highest value of WSF, which is superior compared to all other stemmer listed in Table V. Stemmer is also leading than Gujarati and Punjabi inflective languages. Sanskrit hybrid stemmer is also comparable to Assamese stemmer. User interface for developed system is shown in figure 3. Developed stemmer can be used as a one of the module of morphological analysis system.

6. FUTURE WORK

As developed system works best for stem size more than two letter words, Majority of two letter words are over stemmed and those words are actually root word. Because of lacking of resource and unavailability of handcrafted suffix list, exhaustive hand crafted suffix list was not supplied. As a result, resource preparation for Sanskrit handcrafted suffix list will be necessary for hybrid stemmer enhancement. Still, it is requirement to experiment with huge corpus to see the behavior of stemmer. Strength and accuracy was measured for direct assessment method, indirect assessment method is required to test, which may change accuracy for stemmer. Perhaps stemmer requires to be tuned based on its performance. Execution time needs to be checked for huge database. Word length is major concern as Sanskrit is highly inflective and numerous forms are possible by joining word. Figure.4 shows such few examples where more than two partition of word is required.

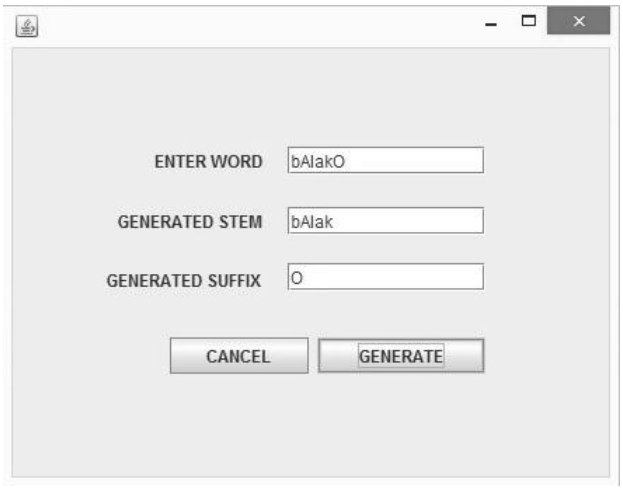

Figure 3: Graphical User Interface of System

References

Amaresh Kumar Pandey, T. J. S. 2008. No Title. In *Proceeding AND '08 Proceedings of the second workshop on Analytics for noisy unstructured text data*. 99–105.

Ameta, J., Joshi, N., and Mathur, I. 2011. A Lightweight Stemmer for Gujarati. In *In Proceedings of 46th Annual National Convention of Computer Society of India.*

Bhadra, M., Singh, S. K., Kumar, S., Subash, Agrawal, M., Chandrasekhar, R., Mishra, S. K., and Jha, G. N. 2009. Sanskrit Analysis System (SAS). In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 1–20.

Bhamidipati, N. L. and Pal, S. K. 2007. Stemming via distribution-based word segregation for classification and retrieval. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 37,* 2, 350–360.

Brigs, R. 1985. Knowledge Representation in Sanskrit and Artificial Intelligence. *THE AI Megazine 6,* 1, 32–39.

Caumanns, J. 1999. A Fast and Simple Stemming Algorithm for German Words. *Technial Reports B 99/16*, 10.

Dolamic, L. and Savoy, J. 2009. Indexing and stemming approaches for the Czech language. *Information Processing & Management 45,* 6 (nov), 714–720.

Goldsmith, J. 2001. Unsupervised Learning of the Morphology of a Natural Language. *Computational Linguistics 27,* 2 (jun), 153–198.

Goyal, P., Huet, G., Kulkarni, A., Scharf, P., and Bunker, R. 2012. A Distributed Platform for Sanskrit Processing. In *Proceedings of COLLING 2012: Techncial papers.* mumbai, 1011–1028.

Hammarström, H. and Borin, L. 2011. Unsupervised Learning of Morphology. *Computational Linguistics 37,* 2 (jun), 309–350.

Huet, G. 2009. Formal structure of sanskrit text: Requirements analysis for a mechanical sanskrit processor. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).*

Jha, G. N., Agrawal, M., Subash, Mishra, S. K., Mani, D., Mishra, D., Bhadra, M., and Singh, S. K. 2009. Inflectional morphology analyzer for sanskrit. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).*

Kumar, D. and Rana, P. 2010. Design and Development of a Stemmer for Punjabi. *Interna-*

tional Journal of Computer Applications 11, 12, 18–23.

Lovins, J. B. 1968. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics 11*, 22–31.

Majumder, P., Mitra, M., and Pal, D. 2008. Bulgarian, Hungarian and Czech Stemming Using YASS. In *Advances in Multilingual and Multimodal Information Retrieval*. Vol. 5152. Springer Berlin Heidelberg, Berlin, Heidelberg, 49–56.

Mayfield, J. and McNamee, P. 2003. Single n-gram stemming. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval - SIGIR '03*. Vol. 1. ACM Press, New York, New York, USA, 415–416.

McNamee, P. and Mayfield, J. 2007. N-Gram Morphemes for Retrieval. *Working Notes for the CLEF 2007 Workshop, 19-21 September, Budapest, Hungary*.

Nehar, A., Ziadi, D., Cherroun, H., and Guellouma, Y. 2012. An efficient stemming for Arabic Text Classification. In *2012 International Conference on Innovations in Information Technology, IIT 2012*. Abu Dhabi, 328–332.

Paice, C. 2006. Stemming. In *Encyclopedia of Language & Linguistics*, K. Brown, Ed. Elsevier, 149–150.

Paik, J. H., Mitra, M., Parui, S. K., and Järvelin, K. 2011. Gras. *ACM Transactions on Information Systems 29,* 4 (nov), 1–24.

Porter, M. 2001. Snowball: A language for stemming algorithms.

Porter, M. F. 1980. The Porter Stemmer Algorithm. *14,* 3, 130–137.

Ramanathan, A. and Rao, D. D. 2003. A Lightweight Stemmer for Hindi. In *In Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL), on Computatinal Linguistics for South Asian Languages*. BU.

Saharia, N. 2010. A Suffix-based Noun and Verb Classifier for an Inflectional Language. 19–22.

Saharia, N., Konwar, K. M., Sharma, U., and Kalita, J. K. 2013. *An Improved Stemming Approach Using HMM for a Highly Inflectional Language*.

Sheth, J. R. and Patel, B. C. 2012. Stemming Techniques and Naïve Approach for Gujarati Stemmer. In *nternational Conference in Recent Trends in Information Technology and Computer Science (ICRTITCS - 2012) Proceedings published in International Journal of Computer Applications*. IJCA, chennai, 975–8887.

Smirnov, I. 2008. Overview of stemming algorithms. *Mechanical Translation*, 1–8.

Suba, K., Jiandani, D., and Bhattacharyya, P. 2011. Hybrid Inflectional Stemmer and Rule-based Derivational Stemmer for Gujarati. In *Proceedings of the 2nd Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), IJCNLP*. Chiang Mai, 1–8.

Prof. Miral Patel Completed her Bachelor of Computer Engineering from Birla Vishwakarma Mahavidyalaya, Sardar Patel University, Gujarat, India and Master of Information Technology from G H.Patel College of Engineering, Gujarat Technological University. Currently working toward the Ph.D. Degree Changa University, Gujarat, India. More than more than 9 Years of Experience in Induary and academic insititues. She has visited USA Gate government Project, Chicago, IL. Her Area of interest includes Natural language processing, Software Engineering, Project management, Artificial intelligence.

Dr. Apurva Shah is working as Associate Professor at Dept. of Computer Science and Engineering, The Maharaja Sayajirao University of Baroda, Vadodara, India. He is also servings as Director, Computer Center, The Maharaja Sayajirao University of Baroda. Prof. Shah did his PhD from Sardar Patel University, India. He is having over 15 years experience of teaching undergraduate and post-graduate students of Computer Engineering and Information Technology. He has published more than 19 International Journal and Conference papers. He visited several countries like USA, Canada, Hong Kong and Singapore for academic and research purpose. He has worked as Chair, Technical Program Committee, and IEEE ISSP 2013. He has also worked as Editor in Chief, Proceedings of the multi-Conference 2011- ICSSA 2011 & ICISD 2011, Brown Walker Press, 2011. He has served as reviewer of reputed journals and conferences. His research interests include Real-Time Systems, Distributed Systems and Artificial Intelligence.