FIRE 2012 Working Notes: Morpheme Extraction Task Using Mulaadhaar – A Rule-based Stemmer for Bengali

Sandipan Sarkar[#1], Sivaji Bandyopadhyay*[2]

[#]*IBM India*
Kolkata, West Bengal, India
[1]sandipansarkar@gmail.com, sandipan.sarkar@in.ibm.com

[*]*Computer Science and Engineering Department, Jadavpur University*
Kolkata, West Bengal, India
[2]sbandyopadhyay@cse.jdvu.ac.in, sivaji_cse_ju@yahoo.com

Abstract— **This paper reports the outcome of the participation of our rule-based stemmer for Bengali in the Morpheme Extraction Task in FIRE 2012. The linguistic rule-based stemmers were largely missing in Bengali language. The proposed system, Mulaadhaar, tried to fill in that space. The accuracy of the tool reached up to 96% in internal evaluation on classic literature and contemporary travelogue domains. In the FIRE 2012 competition, the tool improved the performance of the baseline unstemmed IR system by 20% and achieved the highest MAP of 0.3307 across all tools in Bengali and other languages.**

Keywords— **Stemmer, POS, Information Retrieval, Bengali, Natural Language Processing.**

I. INTRODUCTION

Stemming systems and algorithms were being studied for English being studied since 1968 ([2]). Other languages from Germanic, Italic, Austronesian, Balto-Slavic, and Semitic families were also studied extensively in 1990s and later. However, studies for Indo-Iranian family, where Indian languages belong to, started to appear only from the first decade of the new millennium.

A. Literature Review on Indic Stemmers

Larkey et. al. ([5]) developed a simple suffix stripper stemmer for Hindi based on a list of 27 common suffixes. They reported improvement on average precision of the IR system after applying the stemmer.

Ramanathan et. al. ([6]) reported a rule based affix removal stemmer for Hindi. They reported the accuracy to be 82%.

Dasgupta et. al. ([7]) proposed a Bengali rule-based morphological analyser covering both derivational and inflectional morphology. They applied a lexicon, morphotactics in the form of FSA and orthographic rules for spelling changes. They also applied a morpheme tree structure to guess the final POS of the word formed by the analysed morphemes. Along with other morphemes – the analyser also found the lemma of the word. The work did not report any performance measure of the morphological analyser.

Bhattacharya et. al. ([8]) discussed morphological analysis approaches for Bengali noun, pronoun and verb parts of speech. They described different attributes (e.g. tense, dialect etc.) for these parts of speech and their effects in Bengali inflectional morphology.

Garain et. al. ([9]) applied stemming in the context of Bengali image document retrieval system. The proposed approach was unsupervised clustering based on edit distance (Levenshtein, [1]) of two words. Once clusters are formed, stem of the cluster was identified as the longest substring common to all words in the cluster. The experiment was run on images of newspaper articles and achieved the stemming accuracy of 88.77%.

Dasgupta et. al. ([10]) devised an unsupervised algorithm for any natural language text to induce the prefix, suffix and stems from an unannotated corpus without any prior morphotactics and morpho-phonological rules. The same algorithm was extended to detect composite suffixes. They reported an accuracy of 64.62% while the algorithm was run on Bengali text corpus.

Islam et. al. ([13]) designed a suffix stripper stemmer for Bengali language and applied it in spell checking application.

Majumder et. al. ([12]) took a clustering approach to solve the stemming problem. The clusters were formed from the corpus based on the distance between two words. They argued that Levenshtein edit distance ([1]) may not be appropriate for this purpose and thus proposed four different distance functions, which put weights on mismatches based on the character position in a decreasing manner. The nucleus of the cluster became the stem. For French this approach produced comparable results with respect to Porter-like ([3]) stemmer. For Bengali, in absence of a Porter-like stemmer, they showed that it significantly improved over no-stemming approach.

Pandey et. al. ([13]) proposed an unsupervised approach to identify the stem based on Goldsmith's model ([4]). It calculated the probability of the split of the word based on all combinations of possible stem and inflection combination. It iterated the split probability based on a naïve Bayesian model. The split with maximum probability identified the stem. The accuracy for Hindi language was reported between 85% - 89%, which outperformed both Ramanathan et. al. ([6]) and Larkey et. al. ([5]).

Shrivastava et. al. ([14]) applied a simple stemmer to a Hidden Markov Model (HMM) POS Tagger for Hindi language. To minimize the learning of the HMM, stemmer was applied to expand the input training text into stem and inflections thereby reducing the number of symbols. The result of applying the stemmer achieved the POS tagger accuracy around 93%, which is a 10% improvement on a tagger without the stemmer.

Akram et. al. ([15]) argued that statistical stemmers produce low accuracy in morphologically rich languages, whereas rule-based approach works best for them. They designed a rule-based stemmer for Urdu that strips the prefix and suffixes from the word to produce a meaningful stem, if the word and its stem not successfully found in a lexicon. They reported an accuracy of 91% as a result of the experimentation.

Like in Garain et. al. ([9]), Das et. al. ([16]) presented a clustering based stemming technique for Bengali. However, they applied this technique on text instead of image. The clusters were formed based on minimum edit distance ([1]) based K-means clustering. The accuracy of the stemmer was reported to be 74.06%.

Dolamic ([17]) presented a rule based stemmer to improve the IR performance for three Indic languages – Hindi, Marathi and Bengali. It was argued that the complexity of the stemmer increased with the morphological complexity of the language. In this language set, Bengali was the most complex language, which demanded the most number of rules. Experimental results showed that the IR retrieval performance enhanced by 28%, 42% and 20% for Hindi, Marathi and Bengali respectively.

Patel et. al. ([18]) followed a hybrid approach to come up with a stemmer for Gujarati. The statistical unsupervised approach proposed by Goldsmith et. al. ([4]) was adopted, however, a hand-crafted suffix list was used to better understand the stem and inflection split probabilities. The accuracy of the stemmer was reported to be 67.86%, which received a 17% accuracy boost because of hand-crafted suffix.

Gupta et. al. ([19]) proposed a rule based stemmer for Punjabi nouns and proper names. The accuracy was reported to be 87.37%.

Kumar et. al. ([20]) proposed a suffix stripper stemmer for Punjabi. It achieved the average accuracy of 81.27%.

Suba et. al. ([21]) improved the stemmer reported by Patel et. al. ([18]) by including a POS tagger as a pre-processor and a set of substitution rule. These two modules helped the stemmer to boost the performance by 9.7% and 12.7% respectively while achieving the overall performance at more than 90%.

Chaupattnaik et. al. ([22]) designed a suffix stripper stemmer for Oriya language using Finite State Transducers (FST). They noted the accuracy of the stemmer to be 88%.

From the above literature survey, it is evident that rule-based stemmers were largely missing for Bengali.

B. Observation on Bengali Stemmers

From the above literature survey, we found statistical stemmers for Bengali ([9], [10], [11], [12], and [16]) that all depend on training corpus thus depend on contextual information of the surface word to be stemmed. We found three rule based stemmers by Islam et. al. ([11]), and Dolamic ([17]). Out of these, [11] and [17] took simpler suffix stripping approach, which usually are not very effective. None of them reported their respective stemmers' accuracy. The above analysis shows that a linguistic rule-based stemmer is still missing for Bengali language.

C. Objective

Based on the above literature survey and observations, we set the following objectives for this work:

- Develop a stemmer for Bengali based on linguistic rules.
- Compute and compare its result against multiple domains of Bengali language.
- Compute and compare its performance against similar tools on a baseline system.

II. The Stemmer

A. Key Concepts

We define and present few key concepts here as they formed the building blocks of the subsequent discussions:

O-syllable. Using regular expression syntax, an o-syllable can be represented as where C is a consonant, V is a vowel and D is a diacritic mark or halant. If one or more consonants are present, the vowel becomes a dependent vowel sign [maatraa]. The o-syllabic length $|\tau|$ of token (τ) can be defined as the number of o-syllables in τ.

Verb Stem Type. The verb stems can be categorised in to four classes. These classes exhibit different characteristics while being inflected. The definition is provided below:

TABLE I
VERB STEM TYPE DEFINITION

Class	Identification Characteristics
I	If n = 1. Example: খা [*khaa*], দে [*de*] etc.
II	If n > 1 and the n-th o-syllable has halant as diacritic mark. Only this class of verb stems can have halant at the last o-syllable. Example: কর্ [*kar_*], শেখ্ [*shekh_*] etc.
III	If (n > 1) and (λ = 1) and (vowel of the n-th o-syllable is আ [*aa*]) and (vowel of the first syllable is not ঔ [*ou*]). Example: করা [*karaa*], শেখা [*shekhaa*] etc.
IV	If (n > 1) and ((λ > 1) or ((λ = 1) and (vowel of the first syllable is ঔ [*ou*]))) and (vowel of the n-th o-syllable is আ [*aa*]). Example: আটকা [*aaT_kaa*], ধমকা [*dham_kaa*], দৌড়া [*dourhaa*] etc.

where n = o-syllabic length of the stem, $\lambda = \sum_{j=2}^{n} c_j$ and c_j is the number of consonants in j-th o-syllable of the stem.

Stemming Rules. The stemming rules in Mulaadhaar takes the following form: $\sigma i \rightarrow \sigma$, where σ is the stem of the token and *i* is the inflection.

For verbs additional logic is needed since the verb inflections may affect the stems by changing the vowels of first and last o-syllable. The stemming rules for verbs are presented as a 5-tuple:

$$(L_1, R_1, L_n, R_n, i)$$

where

- L_1 is the vowel of the first o-syllable of post-inflection stem
- R_1 is the vowel of the first o-syllable of actual stem
- L_n is the vowel of the last (n-th) o-syllable of post-inflection stem
- R_n is the vowel of the last (n-th) o-syllable of actual stem
- i is the inflection

These rules are specific to verb stem classes.

Lexicon. Mulaadhaar performance can be further enhanced by consulting and matching the stem against a lexicon. However, considering the complex nature of Bengali script, instead of using the standard edit-distance measure (Levenshtein, 1966), the matching algorithm would be based on weighted edit-distance (WED) measure. In WED, the cost of insertion (C_i), deletion (C_d) and substitution (C_s) would be following:

$$C_i(\gamma) = C_d(\gamma) = \begin{cases} 1, \text{ if } (\gamma \in CO) \text{ or } (\gamma \in VO) \\ 0.5, \text{ if } (\gamma \in VS) \\ 0.25, \text{ if } (\gamma \in DC) \\ 0, \text{ otherwise} \end{cases}$$

$$C_s(\gamma_1, \gamma_2) = \begin{cases} 0, \text{ if } (\gamma_1 = \gamma_2) \\ Min(C_i(\gamma_1), C_i(\gamma_2)), \text{ otherwise} \end{cases}$$

where VO, CO, VS and DC be the set of vowels, consonants, dependent vowel signs and diacritic marks; γ is the character to be inserted or deleted; and character γ_1 is to be substituted by character γ_2.

Stem Ranking. Same token may be generated from multiple stems. While Mulaadhaar returns these stems it should provide some ranking so that the client program can decide which stem to pick up. The rank of the stem would be dependent on three factors, which all would require normalization:

- length of the inflection;
- strength of the rule applied to derive the stem; and
- degree of match of the stem against lexicon entries.

Let $Z = (z_1, z_2, \ldots z_m)$ be the set of stem and associated attributes for surface word w returned by the Stemming Engine. We want to calculate the rank of the stem σ of z_j. Let $i = I(z_j)$ i.e. inflection of z_j, and ρ be rule applied to derive σ.

We define λ_i, the normalized length of inflection i as:

$$\lambda_i = \frac{|i|}{\underset{k=1}{\overset{m}{Max}}(|I(z_k)|)}$$

The normalized strength of rule $\rho = (L_1, R_1, L_n, R_n, i)$ applied to the stem σ is defined as follows:

$$S_\rho = \begin{cases} 1, \text{if } \sigma \text{ is not verb} \\ 1, \text{ if } L_1 \neq \phi \text{ and } L_n \neq \phi \\ 0.5, \text{ if } L_1 \neq \phi \text{ and } L_n = \phi \\ 0.5, \text{ if } L_1 = \phi \text{ and } L_n \neq \phi \\ 0, \text{ otherwise} \end{cases}$$

Let $D = (w_1, w_2, \ldots w_M)$ be a lexicon of size M. The stem σ is compared against D to find near matches by computing WED. Let θ be the threshold value WED is checked against. We define the normalized degree of match as follows:

$$D_\sigma = \frac{\theta - \eta_\sigma}{\theta}$$

where

$$\eta_\sigma = Min(\theta, \underset{k=1}{\overset{M}{Min}}(WED(\sigma, w_k)))$$

We define the ranking formula as below:

$$R_\sigma = \frac{\lambda_i + S_\rho + D_\sigma}{3}$$

B. Stemmer Architecture

We christened this system Mulaadhaar[1]. The structure of Mulaadhaar is provided below using Architecture Description Standard notation ([23]):

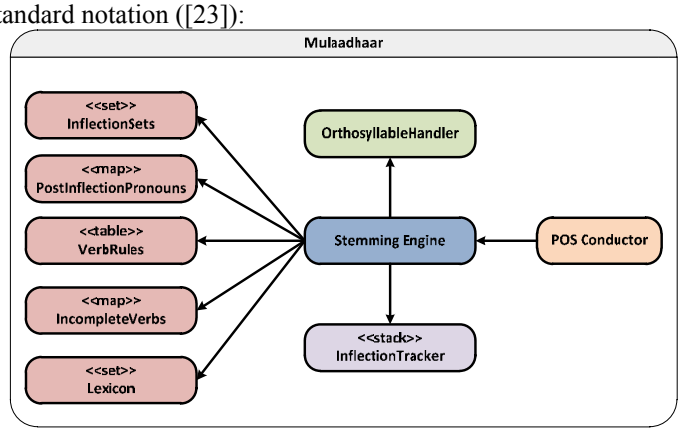

Figure 1: Mulaadhaar Architecture

Mulaadhaar takes surface words as input and produce a set of potential stems along with their possible POS sorted by ranks.

The components of the system are briefly described below:

- **POSConductor**: For each token, it keeps on trying to stem assuming different POS by invoking StemmingEngine. It accumulates the results along with POS and returns the resultant set sorted by rank
- **StemmingEngine**: It receives a tagged token and produces a set of candidate stems with their assigned ranks and associated inflection.
- **OrthosyllableHandler**: This component is responsible for converting a token into o-syllables and vice-versa. It

[1] *Muladhaar (Mulādhār)*, a Sanskrit word, means the "container of root".

also allows calculating the WED between two Bengali tokens.

- **InflectionTracker**: While discovering the inflections recursively, this stack will help the Stemming Engine to keep track of the inflections discovered till now.
- **InflectionSets**: Contains the POS group specific inflection sets.
- **PostinflectionPronouns**: A map of post-inflection pronoun stems against their corresponding actual stem form.
- **VerbRules**: A table of 5-tuple verb rules along with their verb stem class association.
- **IncompleteVerbs**: A map of incomplete verb tokens against their formal imaginary forms.
- **Lexicon**: The dictionary where a discovered stem will be searched for rank enhancement.

The above design is heavily dependent on persisted rules, rather than hard-coded logic. This will bring in configurability and adaptability to the system for easily accommodating other dialects and languages to be considered in future.

III. Experiment and Result

The most popular two evaluation mechanisms for stemmer are direct accuracy measurement ([6], [9], [10], [13], [14], [15], [16], [18], [19], [20], [22], [39], [46], [47], [48], [49]m [50], [51], [52], [53], [54], [55], [56], and [57]) and evaluation using an Information Retrieval (IR) system ([5], [12], [17], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], and [45]). While FIRE 2012 evaluation adopted the later mechanism, we decided to measure the first before submission.

A. Internal Accuracy Measurement

We ran Mulaadhaar on two corpora:

Classic Literature Corpus (CLC). This corpus comprised of first five short stories (ঘাটের কথা [*ghaaTer kathaa*], রাজপথের কথা [*raajapather kathaa*], মুকুট [*mukuT*], দেনাপাওনা [*denaapaaonaa*], and পোস্টমাস্টার [*posTamaasTaar*]) by Rabindranath Tagore. It was written in traditional and colloquial dialects. It contained 15,347 tokens.

Contemporary Travelogue Corpus (CTC). This corpus comprised of four travelogues (আমাজনের গাছবাড়ি [*aamaajaner gaachhabaarhi*], বক্সা-জয়ন্তী [*baksaa-jayantee*], বনসুন্দর [*banasundar*] and বাগান [*baagaan*]) from contemporary travel magazines. It was written in colloquial dialects. It contained 11,561 tokens.

The calculated the performance against two exact accuracy measures – A_1 and A_2, as defined below.

Let $T = (t_1, t_2, \ldots t_N)$ be the set of tokens in a corpus of size N, $S = (\sigma_1, \sigma_2, \ldots \sigma_N)$ be the set of truthed stems for those tokens. Let s'_i and s''_i be the best and second-best stems suggested by the proposed stemmer system for token t_i. Then we define

$$A_1 = \frac{\sum_{i=1}^{N} f'(i)}{N}, \text{ where } f'(i) = \begin{cases} 1, \text{ if } \sigma_i = s'_i \\ 0, \text{ otherwise} \end{cases}$$

and $$A_2 = \frac{\sum_{i=1}^{N} f''(i)}{N}, \text{ where } f''(i) = \begin{cases} 1, \text{ if } \sigma_i \in (s'_i, s''_i) \\ 0, \text{ otherwise} \end{cases}$$

The result is presented in Figure 2.

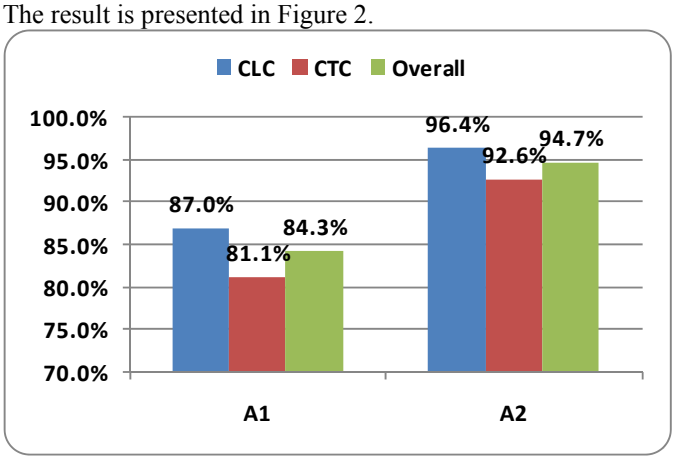

Figure 2: Mulaadhaar Accuracy

B. Evaluation Using Information Retrieval System

Morpheme Extraction Task competition of FIRE 2012 measured the performance of the submitted stemmers using an Information Retrieval system. Our system was designed for Bengali and the test corpus comprised of 1,315,529 tokens. Minimum Average Precision was chosen as the performance metric to compare the results of submitted stemmers. Six systems were submitted to compete in Bengali language. One of them was disqualified because of invalid output format. The results are also compared with the Baseline performance. i.e. the performance of the IR system without stemming. Figure 3 depicts the performance comparison:

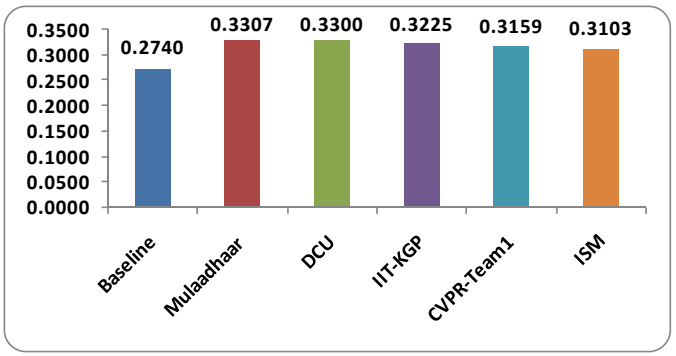

Figure 3: FIRE 2012 Stemmer Performance Comparison

Evidently, all stemmers improved the performance from that of baseline. Mulaadhaar achieved the highest MAP of 0.3307 among all the tools. It enhanced the performance by 20% against baseline. While we compared it with the result of Gujarati, Hindi, Marathi and Odia performances, Mulaadhaar

again achieved the highest score across all systems for these languages.

IV. Conclusion

We designed and developed a rule-based stemmer for Bengali. The evaluation on a classical literature and contemporary travelogue corpus revealed very encouraging results. On our first participation in FIRE competition, we achieved the best performance within Bengali while competing against four other systems. It is a testimonial that the linguistic rule-based methodology can be a successful route for Bengali stemming and information retrieval tasks.

As next steps, it would be interesting to see whether the same system can be used to identify the POS of the surface word by looking at its inflections. Additionally, since Bengali inflections carry the morphological connotations such as person, tense, case etc. it is worth exploring if the same system can be extended as a morphological analyser.

References

[1] V. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals", *Soviet Physics Doklady 10: 707–10*, 1966.

[2] J. V. Lovins, "Development of a Stemming Algorithm," *Mechanical Translation and Computational Linguistics,* vol.11, nos.1 and 2, March and June 1968.

[3] M. F. Porter, "An algorithm for suffix stripping", *Program 14(3):130-137*, 1980.

[4] J. A. Goldsmith, D. Higgins, and S. Soglasnova, "Automatic Language-Specific Stemming in Information Retrieval", *Cross-Language Information Retrieval and Evaluation String Processing and Information Retrieval: Lecture Notes in Computer Science*, Volume 2857/2003, pp. 238-251, 2001.

[5] L. S. Larkey, M. E. Connell, and N. Abduljaleel, "Hindi CLIR in Thirty Days", ACM Transaction on Asian Language Information Processing, Vol-2, No. 2, pp. 130-142, 2003.

[6] A. Ramanathan, and D. D. Rao, "A Lightweight Stemmer for Hindi", *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, 2003.

[7] S. Dasgupta, and M. Khan, "Feature unification for morphological parsing in Bangla", *Proceedings of the 7th International Conference on Computer and Information Technology*, 2004.

[8] S. Bhattacharya, M. Choudhury, S. Sarkar, and A. Basu, "Inflectional Morphology Synthesis for Bengali Noun, Pronoun and Verb Systems", *Proceedings of the National Conference on Computer Processing of Bangla (NCCPB 05)*, pp. 34-4, 2005.

[9] U. Garain, and A. K. Datta, "An approach for stemming in symbolically compressed Indian language imaged documents", *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, 2005.

[10] S. Dasgupta, and V. Ng, "Unsupervised Morphological Parsing for Bengali", *Language Resources and Evaluation*, Volume 40, Numbers 3-4, 311-330, 2006.

[11] M. Z. Islam, M. N. Uddin, and M. Khan, "A Light Weight Stemmer for Bengali and Its Use in Spelling Checker", *Proceedings of the 1st International Conference on Digital Communications and Computer Applications (DCCA2007)*, 2007.

[12] P. Majumder, M. Mitra, S. Parui, and G. Kole, "YASS: Yet another suffix stripper", *ACM Transactions on Information Systems (TOIS),* 2007.

[13] A. K. Pandey, and T. J. Siddiqui, "An Unsupervised Hindi stemmer with heuristic improvements", *Proceedings of the second workshop on Analytics for noisy unstructured text data (AND'08),* 2008.

[14] M. Shrivastava, and P. Bhattacharyya, "Hindi POS Tagger Using Naive Stemming: Harnessing Morphological Information Without Extensive Linguistic Knowledge", *Proceedings of International Conference on NLP (ICON08)*, 2008.

[15] Q. Akram, A. Naseer, and S. Hussain, "Assas-Band, an affix-exception-list based Urdu stemmer", *Proceedings of the 7th Workshop on Asian Language Resources (ALR7)*, 2009.

[16] A. Das, and S. Bandyopadhyay, "Morphological Stemming Cluster Identification for Bangla", *Knowledge Sharing Event-1: Task 3: Morphological Analyzers and Generators*, 2010.

[17] L. Dolamic, "Comparative Study of Indexing and Search Strategies for the Hindi, Marathi and Bengali Language", *ACM Transactions on Asian Language Information Processing (TALIP)*, Volume 9, Issue 3, 2010.

[18] P. Patel. K. Popat, and P. Bhattacharyya, "Hybrid Stemmer for Gujarati", *Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), The 23rd International Conference on Computational Linguistics (COLING),* pp. 51–55, 2010.

[19] V. Gupta, and G. S. Lehal, "Punjabi Language Stemmer for nouns and proper names", *Proceedings of the 2nd Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), IJCNLP 2011*, pp. 35–39, 2011.

[20] D. Kumar, and P. Rana, "Stemming of Punjabi Words by Using Brute Force Technique", International Journal of Engineering Science and Technology (IJEST), Vol. 3 No. 2, pp. 1351-1358, ISSN : 0975-5462, 2011.

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

[22] S. Chaupattnaik, S. S. Nanda, and S. Mohanty, "A Suffix Stripping Algorithm for Odia Stemmer", *International Journal of Computational Linguistics and Natural Language Processing*, Volume 1, Issue 1, 2012.

[23] R. Youngs, D. Redmond-Pyle, P. Spaas, and E. Kahan, "A standard for architecture description", *IBM System Journal*, 38(1), 1999.

[24] M. Lennon, D. S. Peirce, B. D. Tarry, and P. Willett, "An evaluation of some conflation algorithms for information retrieval", *Journal of the American Society for Information Science*, 43(5):384–390, 1991.

[25] D. Harman, "How Effective Is Suffixing?", *Journal of the American Society for Information Science*, 1991.

[26] M. Popovič, and P. Willett, "The effectiveness of stemming for natural-language access to Slovene textual data", *Journal of the American Society for Information Science*, 43(5):384–390, 1992

[27] R. Krovetz, "Viewing Morphology as an Inference Process", *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '93)*, 1993.

[28] W. Kraaij, and R. Pohlmann, "Porter's stemming algorithm for Dutch", *In Noordman LGM and de Vroomen WAM*, eds. Informatiewetenschap 1994: Wetenschappelijke bijdragen aan de derde STINFON Conferentie, pp. 167–180., 1994.

[29] D. A. Hull, "Stemming Algorithm - A Case Study for Detailed Evaluation", *Journal of the American Society for Information Science (JASIS),* 46(9), 1996.

[30] J. Carlberger, H. Dalianis, M. Hassel, and O. Knutsson, "Improving precision in information retrieval for Swedish using stemming", *Proceedings of 13th Nordic conference on computational linguistics (NODALIDA '01)*, 2001.

[31] C. G. Figuerola, R. Gómez, A. F. Z. Rodríguez, and J. L. A. Berrocal, "Stemming in Spanish: A First approach to its impact on information retrieval", *Results of the CLEF 2001 Cross-Language System Evaluation Campaign*, Working Notes for the CLEF 2001 Workshop. 3 September, Darmstadt, Germany. pp. 197-202, 2001.

[32] M. Tashakori, M. Meybodi, and F. Oroumchian, "Bon: The Persian Stemmer", *Proceedings of the First EurAsian Conference on Information and Communication Technology (EurAsia-ICT '02)*, 2002.

[33] M. Aljlayl, and O. Frieder, "On Arabic Search: Improving the Retrieval Effectiveness via a Light Stemming Approach", *Proceedings of the Eleventh International Conference on Information and Knowledge Management (CIKM '02)*, 2002.

[34] C. Monz, and M. de Rijke, "Shallow Morphological Analysis in Monolingual Information Retrieval for Dutch, German and Italian",

Evaluation of Cross-Language Information Retrieval Systems: Lecture Notes in Computer Science, Volume 2406/2002, 1519-1541, 2002.

[35] H. Sever, and Y. Bitirim, "FindStem: Analysis and Evaluation of A Turkish stemming algorithm", *String Processing and Information Retrieval: Lecture Notes in Computer Science*, Volume 2857/2003, pp. 238-251, 2003.

[36] F. Z. Tala, "A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia", Master's thesis, University of Amsterdam, 2003.

[37] C. Jordan, J. Healy, and V. Keselj, "Swordfish: Using Ngrams in an Unsupervised Approach to Morphological Analysis", *Proceedings of Morpho Challenge*, 2005.

[38] T. M. T. Sembok, "Word Stemming Algorithms and Retrieval Effectiveness in Malay and Arabic Documents Retrieval Systems", *Proceedings of World Academy of Science, Engineering and Technology*, 2005.

[39] H. Hammarström, "Poor Man's Stemming: Unsupervised Recognition of Same-Stem Words", *Information Retrieval Technology: Lecture Notes in Computer Science*, Volume 4182/2006, 323-337, 2006.

[40] H. M. Harmanani, W. T. Keirouz, and S. Raheel, "A Rule-Based Extensible Stemmer for Information Retrieval with Application to Arabic", *The International Arab Journal of Information Technology*, Vol.3, No.3, 2006.

[41] J. Savoy, "Light Stemming Approaches for the French, Portuguese, German and Hungarian Languages", *Proceedings of the 2006 ACM symposium on Applied computing (SAC '06)*, 2006.

[42] L. S. Larkey, L. Ballesteros, and M. E. Connell, "Light Stemming for Arabic Information Retrieval", *Arabic Computational Morphology: Text, Speech and Language Technology*, Volume 38, Part IV, 221-243, 2007.

[43] L. Dolamic, "Indexing and Stemming Approaches for the Czech Language", *Information Processing & Management*, Volume 45, Issue 6, , pp. 714–720, 2009.

[44] L. Dolamic, "Indexing and Searching Strategies for the Russian Language", *Journal of the American Society for Information Science*, Volume 60(12), pp. 2540-2547, 2009.

[45] C. Fautsch, and J. Savoy, "Algorithmic Stemmers or Morphological Analysis? An Evaluation", *Journal of the American Society for Information Science and Technology*, Volume 60(12), pp. 1616-1624, 2009.

[46] S. Khoja, "APT: Arabic Part-of-speech Tagger", *Proceedings of the Student Workshop at NAACL 200*1, 2001.

[47] G. Minnen, J. Carroll, and D. Pearce, "Applied morphological processing of English", Natural Language Engineering, pp. 207-223, Cambridge University Press, 2001.

[48] T. Gaustad, and G. Bouma, "Accurate Stemming of Dutch for Text Classification", *Language and Computers*, Selected Papers from the Twelfth CLIN Meeting. Edited by M. Theune, A. Nijholt and H. Hondorp , pp. 104-117(14), 2002.

[49] L. S. Indradjaja, and S. Bressan, "Automatic Learning of Stemming Rules for the Indonesian Language", *Proceedings of the 17th Pacific Asia Conference*, pp. 62-68, 2003.

[50] P. Nakov, "BulStem: Design and Evaluation of Inflectional Stemmer for Bulgarian", *Proceedings of Workshop on Balkan Language Resources and Tools*, 2003.

[51] R. Alshalabi, "Pattern-based Stemmer for Finding Arabic Roots", *Information Technology Journal*, 4(1): 38-43, ISSN 1812-5638, 2005.

[52] H. K. Al Ameed, S. O. Al Ketbi, A. A. Al Kaabi, K. S. Al Shebli, N. F. Al Shamsi, N. H. Al Nuaimi, and S. S. Al Muhairi, "Arabic light stemmer: A new enhanced approach", *Proceedings of The Second International Conference on Innovations in Information Technology (IIT'05)*, 2005.

[53] J. H. Brits, "Outomatiese Setswana lemma-identifisering", Master's Thesis. North-West University, Potchefstroom, South Africa, 2006.

[54] G. Ntais, "Development of a Stemmer for the Greek Language", Master's Thesis, Department of Computer and Systems Sciences, KTH-Stockholm University, 2006.

[55] A. A. Argaw, and L. Asker, "An Amharic Stemmer : Reducing Words to their Citation Forms", *Proceedings of the 5th Workshop on Important Unresolved Matters*, pp. 104–110, 2007.

[56] A. A. Sharifloo, and M. Shamsfard, "A Bottom up Approach to Persian Stemming", *Proceedings of the Third International Joint Conference on Natural Language Processing*, 2008.

[57] H. J. Groenewald, "Using Technology Transfer to Advance Automatic Lemmatisation for Setswana", Proceedings of the EACL 2009 Workshop on Language Technologies for African Languages – AfLaT 2009, pp. 32–37, 2009.