DCU@FIRE-2012: Rule-based Stemmers for Bengali and Hindi

Debasis Ganguly, Johannes Leveling, and Gareth J. F. Jones

CNGL, School of Computing, Dublin City University, Dublin 9, Ireland
{dganguly, jleveling, gjones}@computing.dcu.ie

Abstract. For the participation of Dublin City University (DCU) in the FIRE-2012 Morpheme Extraction Task (MET), we investigated a rule based stemming approaches for Bengali and Hindi IR. The MET task itself is an attempt to obtain a fair and direct comparison between various stemming approaches measured by comparing the retrieval effectiveness obtained by each on the same dataset. Linguistic knowledge was used to manually craft the rules for removing the commonly occurring plural suffixes for Hindi and Bengali. Additionally, rules for removing classifiers and case markers in Bengali were also formulated. Our rule-based stemming approaches produced the best and the second-best retrieval effectiveness for Hindi and Bengali datasets respectively.

1 Introduction

Stemming is an important preprocessing step in information retrieval (IR), which involves normalizing inflected words, essentially representing the same concept, to an equivalent representation in the index (called the stem). For example, an ideal stemmer should normalize the words *friend*, *friends*, *friendly*, and *friendship* to the stem *friend*.

Stemming approaches can broadly be classified into two different categories: i) rule-based, where an inflected word is typically normalized by successively removing the suffixes by applying a set of rules, e.g. the suffixes *s*, *ly*, and *ship* can be removed from the inflections to obtain the stem *friend*; ii) corpus-based, where words are grouped into an equivalent set representing the same concept, by utilizing several corpus-wide statistical features such as co-occurrence or individual word-based features such as edit-distance etc.

The advantages of a rule-based stemmer over a corpus based approach are: i) it is much faster because it does not require any pre-processing step on the indexed documents; ii) the corpus based methods are error prone due to under-training on a corpus not large enough for reliable statistical training; and iii) highly frequent proper nouns might lead to stemming errors.

A disadvantage is that the stemming rules may have to be created manually and for each language. Another limitation of a rule-based stemming approach is that for some inflected words, it might not be possible to formulate a generic enough rule for suffix removal. For example, while it is easy to remove the suffix

ship to get the stem *friend*, a simple suffix removal does not suffice to normalize the word *enmity* to *enemy*. Exceptions thus need to be formulated for a set of words. Despite the limitations, the most widely used stemmers in English are the rule-based ones viz. Porter stemmer [1] and the Lovins stemmer [2].

For Indian languages, which are more inflecting in nature than English, the most commonly used stemming approaches have been the corpus-based one [3, 4]. The only reported work involving rule-based stemming for Bengali is [5]. However, there has not been any reported work seeking direct comparison between the stemming approaches i.e. applying the different stemming approaches on the same document collection and query set, and comparing the retrieval effectiveness obtained by each. The Morpheme Extraction Task (MET) incepted in FIRE-2012 is an attempt to achieve a fair and direct comparison between different stemming approaches.

In this paper, we describe our participating systems in the MET-2012. We developed two rule-based stemmers for Bengali and Hindi. The rest of the paper is organized as follows. Section 2 describes the related work which is followed by Sections 3 and 4 describing the rules and how are they applied for Bengali and Hindi. Section 5 evaluates the proposed stemmers by presenting the official results. Finally, Section 6 concludes the paper with directions for future work.

2 Related Work

Stemming approaches can be classified into different categories, e.g. by the results produced by the stemmer (light stemming [6] vs. aggressive stemming [2]) or by the resources used (corpus-based [7] vs. dictionary-based [8]).

The most widely used stemming approach for English is the rule-based Porter stemmer [1], which successively applies rules to transform a word form into its base form. The successive removal of affixes means that words with a recursive morphological structure are reduced to their base form, e.g. words such as *hopelessness* may be reduced to *hope* by removing the suffixes *ness* and *less*.

Light stemming focuses on removing only a few but the most frequent suffixes from word forms. Recently, light stemming has been researched as a less aggressive means to reduce words to their root form. For English, the *s*-stemmer which removes only the *-s*, *-es*, and *-ies* suffixes from words and other light stemming approaches have been proposed (see, for example, [9] and [10]).

YASS is a clustering-based suffix stripper which has been applied to documents in English, French, and Bengali [3]. YASS identifies clusters of equivalence classes for words by calculating distance measures between strings. This stemmer relies on multiple word lists which have to be extracted from documents, i.e. all words starting with the same character have to be collected in the same word list in a scan over all documents.

Xu and Croft [7] use a combination of aggressive suffix removal with co-occurrence information from small text windows to identify stemming classes. This technique is corpus-based and requires little knowledge about the document

language. The original stemmer was developed for a Spanish document collection [7] and shows an increase in recall for Spanish.

Goldsmith [11] identified suffixes employing a minimum description length (MDL) approach. MDL reflects the heuristic that words should be split into a relatively common root part and a common suffix part. Every instance of a word (token) must be split at the same breakpoint, and the breakpoints are selected so that the number of bits for encoding documents is minimal.

Oard, Levow et al. [12] apply the Linguistica tool by Goldsmith [11] to create a statistical stemmer. Suffix frequencies are computed for a subset of 500,000 words in a document collection. The frequencies of suffixes up to a length of 4 were adjusted by subtracting the frequency of subsumed suffixes. Single-character suffixes were sorted by the ratio between their final position likelihood and their unconditional likelihood. Suffixes were sorted in decreasing order of frequency, choosing a cutoff value where the second derivative of the frequency vs. rank was maximized.

3 Bengali Stemmer

We start this section with a brief introduction to the word inflection grammar of Bengali, which is then used to formulate the explicit rules and the methodology devised to apply them in sequence.

Bengali is an Indo-Aryan language spoken by more than 200 million people in Bangladesh and the Indian state of West Bengal. Bengali is a highly inflectional language with frequent compound suffixes which makes it necessary to apply rules in steps. Morphogical affixing in Bengali can be categorized into: a) Inflectional, where the part-of-speech of the inflected word remains unchanged; and b) Derivational, where the part-of-speech of the inflected word changes.

Since nouns, typically due to their higher Inverse Document Frequency (*idf*) values, are more important in IR than other parts-of-speech [13], for inflectional morphology we restrict our investigation to nouns only. Bhattacharya et al. [14] show that noun inflections can grouped into:

i) *Title markers:* These are the titles such as দেবী (“Mrs.”), বাবু (sir) etc. which are added as suffixes to proper nouns.
ii) *Classifier:* Used to denote plurality and specificity of a noun e.g. a root word ছবি (picture) may be inflected as ছবিগুলো (pictures) or ছবিটা (the picture). A classifier can also indicate the gender of a noun e.g.ছাত্র (student) may be inflected to ছাত্রী to particularly denote a female student.
iii) *Case marker:* Used to denote possessive or accusative relations with other words. The possessive case marker for English is the apostrophe character. English does not use accusative markers. An example of an possessive marker is পরিবারের where the suffix ের is added to the root form পরিবার (family) to mean “family’s”.
iv) *Emphasizer:* These markers are used to emphasize the current word e.g.ছবি may be inflected to ছবিই to denote an equivalent of “only a picture” in English.

All of the above suffix types can appear in a word but only in the specified order e.g.ছবিগুলোকেও, where গুলো (a plurality classifier), কে (an accusative marker) and ও (an emphasizer) have been used to derive "'also those pictures" from the root word ছবি. With reference to the above example, we see that for English language IR "also" and "those" can be easily removed since these are stopwords and the trailing "s" which is the plural classifier, can be removed by a simple rule. But in Bengali it is difficult since an English phrase can map to a single word and not normalizing this word to the base form can result in a poor retrieval performance. Title markers if present come before the case markers.

Algorithm 1 shows the algorithm to remove the suffixes for Bengali and Table 1 illustrates a particular case in the control flow of the former. To handle compound suffixes rules are applied in a series of steps.

Algorithm 1 Bengali Suffix Stripper(w)

```
1: len ← len[w]
2: {Drop the emphasizers}
3: if w[len-1] = ও or w[len-1] = ই then
4:    len ← len-1;
5: end if
6: {Drop the classifiers and case markers}
7: x = {তা, টা, টি, টুকু, কে, র, ের, দের ভাবে}
8: while ∃ x: w=w'x do
9:    w ← w'; len ← len[w'];
10: end while
11: {Drop title markers}
12: x = {কারী, শীল, দেবী, বাবু, ভাই}
13: while ∃ x: w=w'x do
14:    w ← w'; len ← len[w'];
15: end while
16: {Drop the plural suffixes}
17: x = {রা, গুলো, গুলি, গুলোতে, গুলিতে}
18: if ∃ x: w=w'x then
19:    w ← w'; len ← len[w'];
20: end if
21: {Drop the derivational suffixes}
22: V = {Bengali Vowels} ∪ {Bengali Matras}∪ {য়}
23: while w[len-1] ∈ V do
24:    len ← len-1
25: end while
26: if len>2 then
27:    w ← w[0...len-1]
28: end if
29: return w
```

Table 1: Rules for simple suffixes with Bengali examples.

Lines	Suffix type	Bengali notation		ITRANS notation	
4	Emphasizer	আধিক্যই	→ আধিক্য	Adhikya[i]	→ Adhikya
4, 19	Emphasizer and plural classifier	মন্ত্রীরাও	→ মন্ত্রী	mantrI[rAo]	→ mantrI
9	Specific classifier	মুখোশটা	→ মুখোশ	mukhosh[TA]	→ mukhosh
9	Possessive case marker	ভারতের	→ ভারত	bhArat[er]	→ bhArat
9	Plural accusative case marker	শিল্পীদের	→ শিল্পী	shilpI[der]	→ shilpI
9, 9	Specific classifier and Possessive case marker	দুনিয়াটার	→ দুনিয়া	duniyA[TAr]	→ duniyA
14	Derivational	স্থিতীশীল	→ স্থিতী	sthitI[shIl]	→ sthitI
14	Title marker	করুনাদেবী	→ করুনা	karunA[debI]	→ karunA
9, 2	Plural accusative case marker and derivational	ভারতীয়দের	→ ভারতীয়	bhAratiya[der]	→ bhAratiya

4 Hindi Stemmer

In Hindi, the inflections are less complex than Bengali and hence can be addressed by a smaller number of rules. For instance, the accusative markers and emphasizers in Hindi instead of forming inflections as in Bengali, appear as separate words. An example in Hindi is "भारतीय को" (to an Indian) instead of "ভারতীয়কে" as in Bengali.

In fact, there are only four cases of noun inflections for Hindi. Table 2 summarises these with examples. To remove the inflections in Hindi, a very simple rule similar to Step 21 of Algorithm 1 was employed. To be more precise, we go on removing Hindi vowels, matras, anusvara and य from the rightmost part of a word until the first consonant is encountered. The algorithm is outlined in Algorithm 2. It is easy to see that the application of Algorithm 2 on the word लड़कियाँ (girls) yields the stem लड़की (girl).

Table 2: Noun inflections in Hindi

Root		Inflected form	
Word	English translation	Word	English translation
लडका	boy	लड़कें	boys (direct plural)
लडका	boy	लड़कों	boys (indirect plural)
लड़की	girl	लड़कियाँ	girls (direct plural)
लड़की	girl	लडकियों	girls (indirect plural)

Algorithm 2 Hindi Suffix Stripper(w)

```
len ← len[w]
{Drop the derivational suffixes}
V = {Hindi Vowels} ∪ {Hindi Matras} ∪ {य, ँ}
while w[len-1] ∈ V do
    len ← len-1
end while
return w
```

5 Evaluation

Algorithms 1 and 2 were implemented in the C programming language as standalone applications. The complete source code was submitted to the MET organizers, who compiled the code to build the executables at their end. Each stemmer executable takes as input a list of words, which is the set of unique words indexed by Terrier[1] for the FIRE-2011 document collection in a particular language. The stemmer executable then generates a bi-column file, each line of which comprises of the original word tab separated by its stemmed form. The stemmed forms were then used to create a separate index and run retrieval against it.

The official results are shown in Table 3. Our runs are named by "DCU". Five official runs were submmitted for the Bengali task, whereas only two were submitted for Hindi. It can be seen that our Bengali rule-based stemmer achieves a performance improvement of 20.69% over the baseline (no stemming). The improvement obtained over the baseline with our rule-based Hindi is less (5.03%). The most likely reason of getting more improvement for Bengali is that Bengali being a language with more complex morphology than Hindi, offers a larger scope of improvement by the stemming process.

Table 3: Official results of the MET-2012 task.

Team	Language	MAP
Baseline	Bengali	0.2740
JU	Bengali	0.3307 (20.69%)
DCU	Bengali	0.3300 (20.44%)
IIT-KGP	Bengali	0.3225 (17.70%)
CVPR-Team1	Bengali	0.3159 (15.29%)
ISM	Bengali	0.3103 (13.25%)
Baseline	Hindi	0.2821
DCU	Hindi	0.2963 (5.03%)
ISM	Hindi	0.2793 (-0.99%)

[1] http://terrier.org/

6 Conclusions and Future Work

Our rule-based stemmers for Hindi and Bengali yielded the best and second best performance gains in retrieval effectiveness, respectively. Future work will involve extending the rules and adding appropriate exceptions.

Acknowledgments

This research is supported by the Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (CNGL) project.

References

1. Porter, M.F.: An algorithm for suffix stripping. Program **14**(3) (1980) 130–137
2. Lovins, J.B.: Development of a stemming algorithm. Mechanical translation and computation **11**(1-2) (1968) 22–31
3. Majumder, P., Mitra, M., Parui, S.K., Kole, G., Mitra, P., Datta, K.: YASS: Yet another suffix stripper. ACM Trans. Inf. Syst. **25**(4) (2007)
4. Paik, J.H., Pal, D., Parui, S.K.: A novel corpus-based stemming algorithm using co-occurrence statistics. In: Proceedings of the SIGIR '11. (2011) 863–872
5. Leveling, J., Ganguly, D., Jones, G.J.F.: DCU@FIRE2010: Term conflation, blind relevance feedback, and cross-language IR with manual and automatic query translation. In: Second Workshop of the Forum for Information Retrieval Evaluation (FIRE 2010), Working Notes. (2010) 39–44
6. Savoy, J.: A stemming procedure and stopword list for general French corpora. Journal of the American Society for Information Science **50**(10) (1999) 944–952
7. Xu, J., Croft, B.: Corpus-based stemming using co-occurence of word variants. ACM transactions on information systems **16**(1) (1998) 61–81
8. Krovetz, R.: Viewing morphology as an inference process. In: SIGIR, ACM (1993) 191–202
9. Harman, D.: How effective is suffixing? Journal of the American Society for Information Science **42**(1) (1991) 7–15
10. Savoy, J.: Light stemming approaches for the French, Portuguese, German and Hungarian languages. In: Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), Dijon, France, April 23–27, 2006, ACM (2006) 1031–1035
11. Goldsmith, J.: Unsupervised learning of the morphology of a natural language. Computational Linguistics **27** (2001) 153–198
12. Oard, D.W., Levow, G.A., Cabezas, C.I.: CLEF experiments at Maryland: Statistical stemming and backoff translation. In: Cross-Language Information Retrieval and Evaluation, Workshop of Cross-Language Evaluation Forum, CLEF 2000, Lisbon, Portugal, September 21–22, 2000, Revised Papers. Volume 2069 of Lecture Notes in Computer Science (LNCS)., Springer (2001)
13. Xu, J., Croft, W.B.: Improving the effectiveness of informational retrieval with Local Context Analysis. ACM Transactions on information systems **18** (2000) 79–112
14. Bhattacharya, S., Choudhury, M., Sarkar, S., Basu, A.: Inflectional morphology synthesis for bengali noun, pronoun and verb systems. In: In Proceedings of the national conference on computer processing of Bangla (NCCPB. (2005) 34–43