Indian Journal of Science and Technology, *Vol 8(27), DOI:10.17485/ijst/2015/v8i27/82943, October 2015*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Punjabi Stemmer Using Punjabi WordNet Database

Rajeev Puri*, R. P. S. Bedi and Vishal Goyal

Punjab Technical University, Kapurthala Road, Jalandhar - 144601, Punjab, India; rpuri@davjalandhar.com

Abstract

Stemming is used as a pre-processing phase in the information retrieval tasks. The stemming process produces linguistically normalized text, which helps in improving the results of information retrieval tasks. In this paper, a revised suffix removal approach with extended set of stripping rules has been discussed for creating a Punjabi language Stemming tool. The stemming algorithm discussed in this paper uses regular expressions for finding suffix matches. The WordNet* database is used here for improving the stemming results.

Keywords: Brute Force, Rule Based Stemming, Punjabi Stemmer, Suffix Stripping, WordNet .

1. Introduction

Stemming may be defined as the process that conflates the morphologically similar terms into a single root word, that can be used for improving the efficiency of the information retrieval process. Consider an example, where the words – ਲੜਕੇ (ladke), ਲੜਕਿਆਂ (ladkiyan), ਲੜਕੋ (ladkon) etc. are mapped to a single stem word - ਲੜਕਾ (ladka). The stemming process in effect reduces the index size of the documents, making the IR process faster. Also, since after stemming, all inflected words are reduced to a single root word, this improves the RECALL. The Punjabi text stemmers developed earlier could stem nouns and proper names only. Stemming method discussed in this paper uses extended rule set, which stems all categories of Punjabi words. For this, a revised suffix list is created after careful manual inspection of the Punjabi word inflections. A named entities database, along with Punjabi WordNet is used here to skip unnecessary stripping of named entities. Also, an exceptions list is built to avoid stripping of words which were identified as false positives after stripping.

2. Stemming Algorithms

The researchers have proposed a number of stemming algorithms for English language starting from Lovins[7], Martin Porter[10], Krovetz[6]. Also some work has been done for Hindi Language (A Lightweight Stemmer for Hindi by Ramanathan and Durgesh D Rao)[11] and Bengali language (Yet Another Suffix Stripper (YASS) by Prasenjit Majumder et al.)[8]. A rule based stemmer for Bengali has also been given by Sandipan Sarkar and Sivaji Bandyopadhyay[12]. Punjabi language stemmer has been developed for stemming nouns and proper names by Vishal Gupta and Gurpreet Singh Lehal[3]. These stemming algorithms are widely categorized as under –

2.1 Brute Force Approach

Suggested by Frakes and Baeza-Yates[2], uses a table lookup method for stemming. A list of all possible stems and their inflections is prepared. While checking for a possible stem, a brute force is performed on this list to find the root word. The approach provides accurate results for the words which are already present in the lookup table and fails to stem the unknown words. Moreover the approach is language dependent.

2.2 Statistical Approach

Suggested by Hafer and Weiss[4], this approach called "Successor Variety Stemmer" identifies the morpheme boundaries based on the available lexicon and decides where the

*Author for correspondence

words should be broken to get a stem. The approach is language independent. Authors suggest that the successor variety sharply increases when a segment boundary is reached by adding more amount of text thereby decreasing the variety of substrings. This information is used for finding the stem. Hafer and Weiss discussed four ways for segmenting the word -

- Cutoff method, wherein some threshold value is selected for successor varieties and a boundary is reached. If the threshold value is too small, then incorrect cuts will be made, if it is too large, then correct cuts will be missed.
- Peak and Plateau method, wherein segmentation is made after a character whose successor variety exceeds that of the character immediately preceding/following it.
- Complete word method, wherein segmentation is done by detecting a similar complete word in the corpus.
- Entropy method, that takes advantage of the distribution of successor variety letters in the corpus. For a system with n events, each with probability p_i, $1 \le i \le n$ the entropy, H, of the system is defined as

$$H = \sum_{i=1}^{n} -p_i \log_2 p_i$$

H is greatest when the events are equi-probable. As the distribution of probability becomes more and more skewed, entropy decreases.

Benno Stein and Martin Potthast[13] evaluated the successor variety stemmer against rule based stemmer and fond that rule based steamers were giving better results as compared to Successor Variety Stemmer.

Adamson and Boreham[1] reported a method of conflating terms called the shared digram method. This method in fact is not a stemming method and is used to calculate the similarity measures (Dice's coefficient) between pairs of terms based on shared unique digrams using the formula –

$$S = \frac{2C}{A+B}$$

where A is the number of unique digrams in the first word, B the number of unique digrams in the second, and C the number of unique digrams shared by A and B. Such similarity measures are determined for all pairs of terms in the database, forming a similarity matrix. Once such a similarity matrix is available, terms are clustered using a single link clustering method.

In another approach, James Mayfield and Paul McNamee[9] proposed to use n-gram tokenization for stemming. By finding and indexing overlapping sequences of n-characters, many benefits of stemming can be achieved without the knowledge of the underlying language. However, the approach is not suitable for languages (for example Arabic) that use infix morphology. Another disadvantage of this approach is the large database of n-grams which slows down the process.

2.3 Affix Removal Approach

Lovins[7] defined the suffix stripping algorithm that uses the longest match first suffix stripping for finding the root word. A set of rules is defined with common possible suffixes. The word to be stemmed is compared against the list and the matching suffixes are dropped from the word resulting in the root word. Lovins also suggested the recursive procedure to remove each order-class suffix from the root word.

Porter[10] suggested an improved method for suffix stripping which was less complex as compared to Lovin's method. However the improvement over the Lovin's results was not much significant.

Both these algorithms are best suitable for less inflectional language like English and their extension to other languages required the entire rule set to be re-defined as per the language.

Majumdar et al.[8] suggested YASS: Yet Another Suffix Stripper that uses a clustering-based approach to discover equivalence classes of root words and their morphological variants. A set of string distance measures are defined, and the lexicon for a given text collection is clustered using the distance measures to identify these equivalence classes. Its performance is comparable to that of Porter's and Lovin's stemmers, both in terms of average precision and the total number of relevant documents retrieved.

Sajjad Khan et al.[5] have discussed a Template Based Affix Stemmer for a Morphologically Rich Language, that not only depends on removing prefixes and suffixes but also on removing infixes from Arabic text.

3. Revised Punjabi Stemmer using WordNet Database

This paper introduces an improvement over the Punjabi language stemmer developed by Gupta-Lehal[3]. For this

purpose, the required corpus was obtained from Punjabi newspaper articles. The following databases were derived from these articles -

3.1 Suffix List

After careful manual inspection of the newspaper articles, a suffix list of total 75 possible suffixes is prepared. A list of corresponding substitution suffixes is also created. The NULL replacement suffix is defined where the only truncation is required to form the root word.

3.2 Named Entities Database

The named entities play an important role in context based information retrieval. The named entities should be skipped during the suffix stripping process as the suffix removal may altogether change the named entity in context. For example, a stripping rule constructed to remove ੀ (Consonant – 'i') from the word ਵਾਪਰੀ => ਵਪਾਰ (vapri => vapr) will also strip ਧਾਮੀ => ਧਾਮ (dhami => dham), which are altogether two different words. A named entities database is constructed with 26,992 names in Punjabi language to serve as the exception list. Majority of these named entities were obtained from the university rolls and news articles.

3.3 Punjabi Dictionary Database

The Punjabi WordNet developed at Centre for Indian Language Technology, IIT Bombay is used to build local Punjabi dictionary database. The WordNet database provided 53,902 distinct Punjabi words. In addition to this, the news articles provided 1,47,942 unique Punjabi words. This database is be used for checking the presence of stemmed word in dictionary. If the stemmed word is present in the dictionary, then there is a greater probability that the stemming is correct.

3.4 Exceptions List

In suffix stripping process, it is observed that sometimes, a suffix stripping rule is important as it correctly stems a good number of words but gives false positive for only a few words. An exceptions list is prepared after inspecting such rules and consequently the words giving false positive are added to the exception list. A threshold value is also assigned to the rule that gives false positive. This threshold value is calculated as a percentage of words giving false positives out of the total words affected by that rule. The algorithm can be tuned to ignore the rules with higher threshold values.

4. Algorithm

1. Read the input string from the Corpus file.
2. Tokenize the Corpus.
3. Remove the Stop Words from tokens.
4. Load the Suffix: Replacement pair sorted on Suffix length into memory with desired threshold level.
5. For each Token W_i in the Token list
   a) If LENGTH (W_i) < MIN. LENGTH OF WORD THEN
   Ignore W_i and proceed with next token.
   b) if (W_i) $\in$ NAMED ENTITIES THEN
   Ignore W_i and proceed with the next token.
   c) if (W_i) $\in$ EXCEPTIONS LIST THEN
   Ignore W_i and proceed with the next token.
   d) For each Suffix S_i in Suffix list
   TEST If $S_i \subseteq W_i$ THEN
   Replace S_i with Replacement R_i in W_i to get possible root word P_i
   e) TEST If $P_i \in$ DICTIONARY
   Record Pi as the root-word

5. Sample Runs

The algorithm was tested on 1000 sample news articles from different domains. The outputs were evaluated for under-stemming and over-stemming errors. Table 1 shows a random sample of input words along with output as stemmed words.

6. Evaluation

The stemming process is said to be effective when the words that are morphological variants are actually conflated to a single stem. An effective stemmer should actually minimize the errors due to over-stemming (a situation where the stemmed words are not actually the morphological variants of the given word) as well as under-stemming (when the morphological words fail to conflate due to lack of corresponding rule).

The stemming results were manually inspected using the GUI tool developed for this purpose. The stemming results – correct stemming and incorrect (under-stemming & over-stemming) were recorded using the facility designed in the tool.

Table 1 Input and stemmed words

Input	Stemmed to word	Input	Stemmed to word
ਵੱਲੋ (valon)	ਵੱਲ (val)	ਜਾਵੇਗੀ (javegi)	ਜਾਵੇ (jave)
ਸੁਣਵਾਈ (sunvaai)	ਸੁਣਵਾ (sunva)	ਅਗਵਾਈ (agvaai)	ਅਗਵਾ (agvaa)
ਆਗੂਆਂ (aaguan)	ਆਗੂ (aagu)	ਜਤਾਈ (jataai)	ਜਤਾ (jataa)
ਟਿਪੱਣੀ (tippani)	ਟਿਪੱ (tipp)	ਸਬੰਧੀ (sambandhi)	ਸਬੰਧ (sambandh)
ਹੋਈਆਂ (hoiyan)	ਹੋਈ (hoi)	ਲੈਣ (len)	ਲੈ (le)
ਗਿਰਫਤਾਰੀਆਂ (giraftarian)	ਗਿਰਫਤਾਰੀ (giraftari)	ਵਜੋ (vajon)	ਵਜ (vaj)
ਰੱਖਦੇ (rakhde)	ਰੱਖ (rakh)	ਕੀਤੇ (kitte)	ਕੀਤਾ (kita)
ਹੋਏ (hoye)	ਹੋ (ho)	ਸ਼ਹਿਰੀ (shehari)	ਸ਼ਹਿਰ (shehar)
ਵੱਲੋ (vallon)	ਵੱਲ (vall)	ਅਧਕਿਾਰੀ (adhikari)	ਅਧਕਿਾਰ (adhikar)
ਸਿਹਮਤੀ (sehmati)	ਸਿਹਮਤ (sehmat)	ਮਨਜੂਰੀ (manjoori)	ਮਨਜੂਰ (manjoor)
ਕਰਦਿਆਂ (kardian)	ਕਰਦਾ (karda)	ਹੋਵੇਗਾ (hovega)	ਹੋਵੇ (hove)
ਖੇਤਰਾਂ (khetran)	ਖੇਤਰ (khetar)	ਪਹਿਲਾਂ (pehlan)	ਪਹਿਲ (pehl)

Precision and recall, the basic measures in IR tasks, were used in evaluating the stemming results as per the following -

6.1 Recall

Recall is the ratio of the number of relevant records retrieved to the total number of relevant records in the database. It is usually expressed as a percentage.

If A represents the number of correctly stemmed words, B represents the number of under-stemmed words then RECALL can be defined as

$$\text{RECALL} = \frac{A}{A+B} \times 100\%$$

6.2 Precision

Precision is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved. It is usually expressed as a percentage.

If A represents the number of correctly stemmed words, C represents the number of over-stemmed words then Precision can be defined as

$$\text{PRECISION} = \frac{A}{A+C} \times 100\%$$

6.3 F-Measure

F-Measure is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score. The F-measure is calculated as –

$$F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

When Precision and Recall are equally important, then $\beta = 1$
and

$$F_1 = \frac{2PR}{P+R}$$

7. Results and Discussion

Table 2 shows the summarized stemming results.

The Table 3 indicates the sample results of first 20 articles (~49000 corpus words) out of total 1000 articles. The high value of F-measure indicates that the algorithm discussed here is highly reliable.

Table 2. Summarised stemming results

Average Precision	90.84%
Average Recall	94.83%
Average F-Measure	92.79%

Table 3. Detailed stemming results for 20 random runs of algorithm

Sr #	Corpus Words	Stemmed Words	Correct Stemmed	Under Stemmed	Over stemmed	Precision	Recall	F-Score
1	179	44	36	2	6	85.71	94.74	90.00
2	290	42	40	2	1	97.56	95.24	96.39
3	246	44	34	4	6	85.00	89.47	87.18
4	268	49	44	1	4	91.67	97.78	94.62
5	209	24	16	2	2	88.89	88.89	88.89
6	140	27	22	2	3	88.00	91.67	89.80
7	284	51	43	5	3	93.48	89.58	91.49
8	273	84	71	3	10	87.65	95.95	91.61
9	544	79	67	6	6	91.78	91.78	91.78
10	268	67	61	2	5	92.42	96.83	94.57
11	151	20	17	1	2	89.47	94.44	91.89
12	251	36	33	1	2	94.29	97.06	95.65
13	93	17	15	1	1	93.75	93.75	93.75
14	111	28	21	2	5	80.77	91.30	85.71
15	170	34	39	0	4	90.70	100.00	95.12
16	216	40	35	1	4	89.74	97.22	93.33
17	244	39	36	0	3	92.31	100.00	96.00
18	220	37	36	1	0	100.00	97.30	98.63
19	312	70	63	2	5	92.65	96.92	94.74
20	432	89	80	1	8	90.91	98.77	94.67

8. References

1. Adamson G, Boreham J. The use of an association measure based on character structure to identify semantically related pairs of words and document titles. Information Storage and Retrieval. 1974 Jul-Aug; 10(7-8):253–60.
2. Frakes WB, Ricardo BY. Information Retrieval: Data Structures and Algorithms. Prentice Hall. 1992.
3. Gupta V, Lehal GS. Punjabi Language Stemmer for nouns and proper names. Proceedings of the 2nd Workshop on South and Southeast Asian Natural Language Processing (WSSANLP). IJCNLP 2011. Chiang Mai. Thailand. 2011 Nov. p. 35–9.
4. Hafer M, Weiss S. Word segmentation by letter successor varieties. Information Storage and Retrieval. 1974 Nov-Dec; 10(11-12):371–85.
5. Khan S, Anwar M, Bajwa U, Xuan W. Template based affix stemmer for a morphologically rich language. The International Arab Journal of Information Technology. 2015 Mar; 12(2):146–54.
6. Krovetz B. Viewing morphology as an inference process. Artificial Intelligence. 2000 Apr; 118(1-2):277–94.
7. Lovins JB. Development of a Stemming Algorithm. Mechanical Translation and Computational Linguistics.1968 Mar & Jun; 11(1-2):22–31.
8. Majumder P, Mitra M, Parui SK, Kole G.YASS: Yet another suffix stripper. ACM Transactions on Information Systems (TOIS). 2007; 25(4):18.
9. Mayfield J, McNamee P. Single n-gram stemming. Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval. 2003. p. 415–6.
10. Porter MF. An algorithm for suffix stripping. Program: electronic library and information systems. 1980; 14(3):130–7.
11. Ramanathan A, Rao DD. A Lightweight Stemmer for Hindi. Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL) on Computational Linguistics for South Asian Languages (Budapest, Apr.) Workshop. 2003; 42–8.
12. Sarkar S, Bandyopadhyay S. Design of a Rule-based Stemmer for Natural Language Text in Bengali. Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages. Hyderabad India. 2008 Jan. p. 65–72.
13. Stein B, Potthast M. Putting Successor Variety Stemming to Work. Proceedings of the 30th Annual Conference of the Gesellschaft für Klassifikation e.V. Freie Universität Berlin. 2006 March 8–10. p. 367–74.