A Rule based approach for lemmatisation of Punjabi text Documents

Dr. Rajeev Puri

Assistant Professor, Dept of Comp. Sc, DAV College, Jalandhar.
rpuri@davjalandhar.com

Keywords: Punjabi lemmatiser, stemming, information retrieval

ABSTRACT

Information retrieval is a trivial task specially when the information is available in highly inflectional languages such as Hindi and Punjabi. The huge corpus needs to be reduced significantly so as to speed up the information retrieval process as well as for generating the accurate results. The pre-processing phase of the information retrieval process deals with reducing the size of corpus. Many pre-processing steps such as stop word removal, stemming and lemmatisation need to be performed for improving the accuracy of information retrieval task. Where the stemming process converts the inflectional forms of word by stripping the prefix or postfix, on the other hand the lemmatisation is the process of generating normalised dictionary word that has the same or related meaning for the given word. The lemmatisation provides accurate results then a stemmer. However Punjabi being an inflectional language makes the task of lemmatisation more complex. Behind each Punjabi world there are many inflectional forms as well as synonyms which need to be dealt with. The Punjabi lemmatiser presented in this text is an attempt to obtain the lemma of the word by using the rule-based approach.

1. Introduction

Lemmatization is process of grouping together the different inflected forms of a word so that they can be analyzed as a single item. It is used for finding out the normalized form of the word or the dictionary form of the word which is called lemma. Lexeme is a group of all the forms (called headwords of a dictionary) that have the same meaning, and lemma

is a word that is chosen as a base form that represents lexeme. Because of the inflectional nature of Punjabi, it makes the task of Lemmatization more complex than Stemming. Both Stemming and Lemmatization are the important preprocessing steps in Information Retrieval task. It is also used in many applications of text mining. It plays an important role in Natural Language Processing and many other linguistics fields. It helps the search engines in finding out the keywords and helps in reducing the size of index files. For building a Lemmatizer one requires proper understanding of the language and dictionary of that particular language. Indian languages are highly inflectional in nature. Punjabi is also highly inflectional language. Various inflectional forms of Punjabi words are shown in table 1 below.

Table 1: Sample Inflectional forms of Punjabi Text

Punjabi Word	Inflectional forms
ਮੁੰਡਾ (Boy)	ਮੁੰਡੇ, ਮੁੰਡਿਆ, ਮੁੰਡਿਆਂ , ਮੁੰਡਿਓ
ਕਰ (Do)	ਕਰਦਿਆਂ, ਕਰਦੀਆਂ, ਕਰੂ, ਕਰੋ, ਕਰਦਾ, ਕਰਦੇ, ਕਰਦੋ
ਠਿਹਰ (Stop)	ਠਹਿਰਾ, ਠਹਿਰੇ, ਠਹਿਰਿਆ, ਠਹਿਰਾਅ
ਰੰਗ (Color)	ਰੰਗਣਾ, ਰੰਗਣ, ਰੰਗੀਨ, ਰੰਗੀਨੀ, ਰੰਗਾਉਣਾ, ਰੰਗਵਾਉਣਾ, ਰੰਗਾਈ, ਰੰਗਵਾਈ, ਰੰਗਲਾ, ਰੰਗੀਲਾ

2. Literature Survey

Taghva et al. [6] have developed Stemmer for Farsi language. Their system uses Deterministic Finite automata for implementation of stemmer. The minimum length of the stem is limited to three words. Sarkar et al. [10] proposed a Rule Based Stemmer for Bengali language which is also a resource poor language. The POS specific rules are applied to find out the stem of the word. Kraai et al. [4] have developed Suffix Stripper for Dutch by using Porter's Algorithm. They extend the original Porter's Algorithm to handle prefixes and infixes. Lovins. [1] proposed a Stemming Algorithm for English. The Algorithm has been developed to be used in Library Information Transfer Systems. Bijal et al. [18] discussed different stemming Algorithms for non-Indian and Indian Languages. They presented a

comparative study of various Stemming Algorithms. Ganguly et al. [16] have developed two rule based stemmers for Bengali and Hindi. Gupta et al. [11] have developed Punjabi Language stemmer for Nouns and Proper Names. First they generate the stem of the Punjabi word and after that stem is checked in Punjabi Noun and Proper Name Dictionary. Hafer et al. [2] have developed word segmentation by letter successor varieties method. They used Statistical properties of the corpus to decide whether word should be divided or not. The approach is based upon Z.S Harris Process. Majumder et al. [9] have developed a Stemmer YASS (yet another suffix stripper) which is based on clustering based approach. Cluster is a class of root word and their related word forms. Puri et al. [19] have developed a Punjabi Stemmer using Punjabi WordNet Database. In this paper Suffix Removal Approach is used with suffix stripping rules for finding out the stem for Punjabi Language. El-Shishtawy et al. [13] have developed an accurate Arabic Root-Based Lemmatizer for Information Retrieval Purpose. This is the first Lemmatizer for Arabic Language based on non-statistical approach. Mishra et al. [14] have developed MAULIK: an effective Stemmer for Hindi based on Hybrid Approach. The Hybrid Approach is combination of Brute Force and Suffix Removal Approach. This Stemmer is based on Devanagari Script.

3. Research Gap

Punjabi is an indo-aryan Language. It is spoken by more than 100 million native people worldwide. It is the 10th most widely spoken Language in the world. The popularity and online presence of Punjabi language is increasing day by day as more and more Punjabi corpus are generated and published day by day over internet. So a system is required for efficient searching of Punjabi documents over the web. The searching should be in native's own language. For faster and efficient information retrieval system in Punjabi, some pre-processing is required before indexing the documents. This pre-processing includes stop word removal, stemming and a sophisticated Lemmatizer for reducing the words to their base forms called headwords. Various Lemmatization and Stemming systems are available for different languages, but very less work has been

done for Punjabi because of its inflectional nature. For every word in Punjabi, there are number of inflectional forms. So, attempt here has been made to develop a Lemmatizer for Punjabi which will generate accurate lemma for most of the Punjabi words.

4. Proposed System

A proposed system uses rule-based approach to find out the normalized form of the word that helps in improving the results of Information Retrieval task. The system works with the Synonym replacement module that uses pre-defined rules to get the lemma of the word.

For e.g:- Word:- ਬਿਮਾਰ

Synonym Words:- , ਰੋਗੀ, ਮਰੀਜ, ਮਰੀਜ਼, ਰੋਗਗ੍ਰਸਤ

The word ਬਿਮਾਰ has length 15, is the input word for the system. The list of synonym is collected from the database and then system picks the shortest length word from the list that is ਰੋਗੀ (12) with length 12. After that the rules from the rule list are applied on the synonym word to get the accurate lemma word as an output. Here, the 'ੀ' => "" rule of suffix stripping will be applied on the word ਰੋਗੀ (12) and it becomes ਰੋਗ (9), and that is the normalized word, also a valid word available in the dictionary and displayed as an output on the output screen. The proposed system uses a number of databases and lists for accomplishing the task. These databases and lists are mentioned in the coming sections.

5. Synonyms Database

Synonyms are the words that have the same meaning. A synonym database is prepared with augmented field containing the length of the word for quick comparison. The database is indexed on the word and sorted according to the length of each word in ascending order for quick retreival. For example, the word "ਮੂਰਖ" (Fool) is having 26 distinct synonyms as "ਜਾਹਲ", "ਉਜੱਡ", "ਜੜ੍ਹ", "ਗਵਾਰ", "ਭੁੱਚ", "ਘੋਗਾ", "ਬੇਸ਼ਮਝ", "ਮੂੜ੍ਹ", "ਅਬੁੱਧ", "ਨਾਸਮਝ", "ਘੁੱਗੂ", "ਘੁੱਘੂ", "ਝੁੱਡੂ", "ਬੇਵਕੂਫ", "ਅਗਿਆਨੀ", "ਮੱਤਹੀਣ", "ਲੱਲ੍ਹਾ", "ਅੰਞਾਣਾ", "ਜੜ_ਮੱਤ", "ਮੂੜ੍ਹਮੱਤ", "ਬੁੱਧੀਹੀਣ", "ਨਿਰਬੁੱਧੀ", "ਲਘੂ_ਬੁੱਧੀ", "ਅਲਪ_ਬੁੱਧੀ", "ਘੱਟ_ਬੁੱਧੀ".

Out of all these distinct synonyms, the synonym “ਜਾਹਲ” has the minimum length of 12 characters and hence is chosen as a replacement of all other peer words. The synonym replacement module has some inherent problems of incorrectly replacing a named entity with it’s synonym. For example, the name “ਪ੍ਰਕਾਸ਼ ਸਿੰਘ ਬਾਦਲ” can be incorrectly transformed to “ਚਾਨਣ ਸ਼ੇਰ ਮੇਘ” . To avoid this type of problems, a named entity recognition is required.

6. Suffix Rules list

Suffix are the inflectional endings of the words in any language. The need of the Suffix Rule List arises in order to remove the inflectional endings of the word. Rules can either remove or add a character at the end of the word in order to get a valid head word. It can either be the inflection of input word or the synonym word. Rules are applied on input words by using the regular expressions. Some of the suffixes used in the system are shown in Table 2 below –

Table 2: Sample Suffix list for stripping

Suffix	Words
ਾਂ	ਦੁਕਾਨਾਂ , ਮਕਾਨਾਂ , ਸ਼ਮਸ਼ਾਨਾਂ
ਕਾਰੀ	ਕਾਰਜਕਾਰੀ , ਹਿਤਕਾਰੀ , ਲਾਭਕਾਰੀ
ਣਗੇ	ਚਲਣਗੇ , ਰੁਕਣਗੇ , ਦੇਖਣਗੇ
ੂਆਂ	ਹੰਜੂਆਂ , ਆਗੂਆਂ
ਗਾ	ਆਵੇਗਾ , ਜਾਵੇਗਾ, ਕਰੇਗਾ , ਭਰੇਗਾ

The list of words having the common suffixes is prepared and frequency of valid suffixes is calculated. The suffix with maximum occurrences is stripped first, followed by suffix with largest length. The suffix stripping also needs to exclude named entities from stripping.

7. Named Entities database

Named entities sometimes create problems in regular lemmatization process. The rules defined for lemmatization may sometimes give a false positive and lemmatize a named entity. As mentioned in the proposed system section, the word ‘ਜੋਸ਼ੀ’ becomes ‘ਜੋਸ਼’ after applying the ‘ੀ’ => “” rule and ‘ਸ਼ਰਮਾ’ becomes ‘ਸ਼ਰਮ’ after appling the ‘ਾ’=> “” rule, where both ‘ਜੋਸ਼’ and ‘ਸ਼ਰਮ’ are valid dictionary words, but with an altogether different meaning. So for avoiding these kind of problems with the named entities, it is required to ignore the

named entities so that only the words that need lemmatization should be lemmatized.

8. Dictionary Database

Dictionary database is the collection of all the valid words. The need of the dictionary arises due to check on the output of the System. If the output word is available in the dictionary then the output is considered as a valid lemma. The bigger is the size of the dictionary, the better it will be in terms of accuracy of the system.

9. Algorithm

The lemmatization algorithm designed is mentioned as below -

```
For each word Wi,
        if (Wi) found in {NAMED ENTITIES} THEN
                Ignore Wi and Continue
        If (Wi) can be stemmed
                Stem Wi
                Check in {Dictionary}
                if (found)
                        replace Wi with dictionary
word.
                else
                        proceed to next step.
        Endif
        If(Wi) found in {Synonyms Database}
                replace Wi with synonym of min
length.
        Endif
        Output Wi as Lemmatised word
EndFor
```

10. Results and Discussions

The system is tested on 50 articles from a popular newspaper. The performance of the system is measured using Precision-Recall and F-Score is calculated based on it. The result from 10 random articles is shown in Table 3.

Table 3: Result from 10 random articles from a list of 50 articles used for testing.

Sr.	Words	TP	FP	FN	P	R	F
1	332	76	0	6	81	93	87
2	380	117	0	7	76	94	84
3	498	124	0	0	80	100	89
4	366	120	7	10	75	92	83
5	347	117	6	10	75	92	83
6	493	138	4	0	78	100	88
7	357	114	7	3	76	97	85
8	385	123	0	0	76	100	86
9	421	101	0	0	81	100	89
10	400	104	8	12	79	90	84

Based on the precision and recall values, the average precision, average recall and average f-score is calculated. The results of averages is shown in Table 4.

Table 4: Average Precision, Recall and F-Score.

Average Precision	78
Average Recall	95
Average F-Score	86

The graphical representation of results of the system is shown in Fig. 1. The graph shows the precision, recall and F-

measure of 50 articles lemmatised using the algorithm above.

Fig. 1 Results from Lemmatisation algorithm.

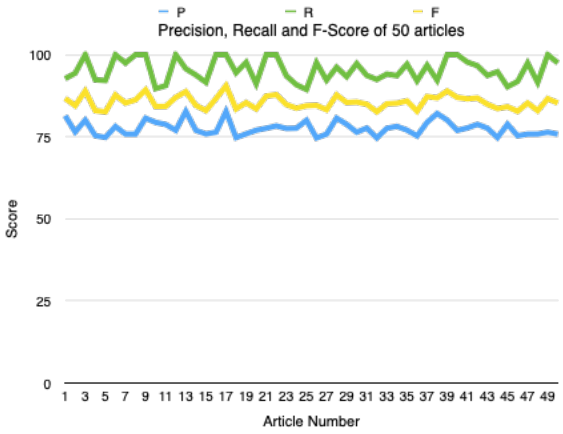
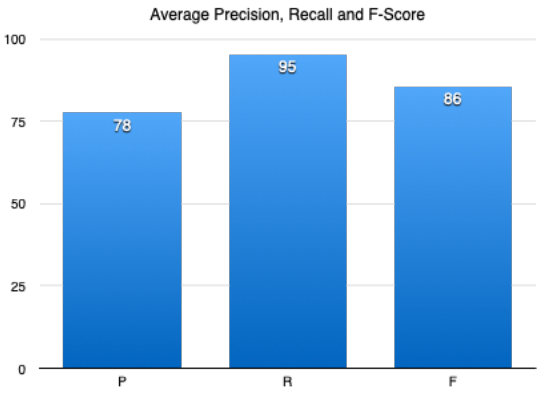


Fig 2 shows the graphical results of average precision, recall and f-score values.

Fig 2. Average Precision, Recall and F-Score.



11. Conclusion

In this paper, the design of rule based Punjabi Lemmatiser is discussed. The rule lists have been generated manually for the development of the system. Firstly, the system finds out the synonym of the word if any, and after that applies the rules for generating the lemma of the word. Various databases of Punjabi words have been used to improve the performance of the system. The system can be beneficial for other applications of Natural Language Processing. The performance of the system can be improved by adding more words into the database. In future the system can be improved by handling the Context and POS (Part of Speech) of the words.

References

1. Lovins, J. B. (1968). Development of a stemming algorithm (p. 65). Cambridge: MIT Information Processing Group, Electronic Systems Laboratory.

2. Hafer, M. A., & Weiss, S. F. (1974). Word segmentation by letter successor varieties. Information storage and retrieval, 10(11), 371-385.

3. Porter, M. F. (1980). An algorithm for suffix stripping. Program, 14(3), 130-137.

4. Kraaij, W., & Pohlmann, R. (1995). Evaluation of a Dutch stemming algorithm. The New Review of Document and Text Management, 1, 25-43.

5. Ramanathan, A., & Rao, D. D. (2003, April). A lightweight stemmer for Hindi. In the Proceedings of EACL.

6. Taghva, K., Beckley, R., & Sadeh, M. (2005, April). A Stemming Algorithm for the Farsi Language. In ITCC (1) (pp. 158-162).

7. Al-Shalabi, R., Kannan, G., Hilat, I., Ababneh, A., & Al-Zubi, A. (2005). Experiments with the successor variety algorithm using the cutoff and entropy methods. Information Technology Journal, 4(1), 55-62.

8. Willett, P. (2006). The Porter stemming algorithm: then and now. Program,40(3), 219-223.

9. Majumder, P., Mitra, M., Parui, S. K., Kole, G., Mitra, P., & Datta, K. (2007). YASS: Yet another suffix stripper. ACM transactions on information systems (TOIS), 25(4), 18.

10. Sarkar, S., & Bandyopadhyay, S. (2008, January). Design of a Rule- based Stemmer for Natural Language Text in Bengali. In IJCNLP (pp. 65-72).

11. Gupta, V., & Lehal, G. S. (2011, November). Punjabi Language Stemmer for nouns and proper names. In proceedings of the 2nd Workshop on South and Southeast Asian Natural Language Processing (WSSANLP) IJCNLP (pp. 35-39).

12. Sharma, D. (2012). Stemming algorithms: a comparative study and their analysis. International Journal of Applied Information Systems, 4(3), 7-12.

13. El-Shishtawy, T., & El-Ghannam, F. (2012). An accurate arabic root- based lemmatizer for information retrieval purposes. arXiv preprint arXiv:1203.3584.

14. Upendra, M., & Chandra, P. (2012). MAULIK: An Effective Stemmer for Hindi Lanuage. International Journal of Computer Science and Engineering, IJCSE, 4(5), 711-717.

15. Saharia, N., Sharma, U., & Kalita, J. (2012, August). Analysis and evaluation of stemming algorithms: a case study with Assamese. InProceedings of the International Conference on Advances in Computing, Communications and Informatics (pp. 842-846). ACM.

16. Ganguly, D., Leveling, J., & Jones, G. J. (2013, December). DCU@ Morpheme Extraction Task of FIRE-2012: Rule-

based Stemmers for Bengali and Hindi. In Post-Proceedings of the 4th and 5th Workshops of the Forum for Information Retrieval Evaluation (p. 12). ACM.

17. Paul, S., Joshi, N., & Mathur, I. (2013). Development of a hindi lemmatizer.arXiv, preprint arXiv:1305.6211.

18. Bijal, D., & Sanket, S. (2014). Overview of Stemming Algorithms for Indian and Non-Indian Languages. arXiv preprint arXiv:1404.2878.

19. Puri, R., Bedi, R. P. S., & Goyal, V. (2015). Punjabi stemmer using punjabi wordnet database. Indian Journal of Science and Technology, 8(27).

20. Aarti, D. V. S. Punjabi Language Characteristics and Role of Thesaurus in Natural Language processing.