DESIGN OF A RULE BASED HINDI LEMMATIZER

Snigdha Paul, Mini Tandon, Nisheeth Joshi and Iti Mathur

Apaji Institute, Banasthali University, Rajasthan, India
snigdha.pal18@gmail.com, minitndn200@gmail.com,
nisheeth.joshi@rediffmail.com,mathur_iti@rediffmail.com

ABSTRACT

Stemming is the process of clipping off the affixes from the input word to obtain the respective root word, but it is not necessary that stemming provide us the genuine and meaningful root word. To overcome this problem we come up with a solution- Lemmatizer. It is the process by which we crave out the lemma from the given word and can also add additional rules to make the clipped word a proper stem. In this paper we have created an inflectional lemmatizer which generates the rules for extracting the suffixes and also added rules for generating a proper meaningful root word.

KEYWORDS

Stemming, Lemmatization, Lemma, Hindi, Over-stemming and Under-stemming.

1. INTRODUCTION

In today's era we live in a computerized world where almost everything is done by computers. With this advent we have a lot of data to store and retrieve. This stored data can be accessed for retrieving information from anywhere. This is a very good application of Natural Language Processing. Since language is so vast and has its own grammar structure, it is very necessary to study the word structure. For example – '*indicate*' can be written as *indicative, indication, indicated, indicator, indicatable*. Here we need some mechanism to drop all the word forms to a common base form which is called as a root word or stem. This is done by stemming or lemmatization. Stemming is the process of reducing the words to a common stem by clipping off the unnecessary morphemes. These morphemes are called as suffixes. This suffix stripping is done by generating various rules. For example – '*introduce*' has many word forms like *introduction, introducing, introduced*. After stemming, all the forms are contracted to the word *introduce*- which is not a proper root word. This problem is resolved by lemmatizer where the inflectional ending is detached and the base or dictionary form of the word is generated. If we find the root word of the above example then we get it as '*introduce*' but not '*introduc*'. Thus in stemming we have two kinds of problem – *Under-stemming* and *Over-stemming*. Under-stemming is a case which occurs when the words need to be grouped together, but are not actually grouped. It occurs mostly in case of semantically same words. For example – in Hindi we have a

Sundarapandian et al. (Eds) : ACITY, AIAA, CNSA, DPPR, NeCoM, WeST, DMS, P2PTM, VLSI - 2013
pp. 67–74, 2013. © CS & IT-CSCP 2013 DOI : 10.5121/csit.2013.3408

word *गलतियों* which when stemmed gives *गलतिय*, but it is not a proper root. Over-stemming is a case which occurs when the words need not to be grouped together, but are actually grouped. It occurs mainly in case of semantically distant words. For example – the Hindi word *चहेता* when stemmed gives us *चहे* which is again not a proper root word. Thus we can say that stemmer does not provide us the contextual knowledge which is provided by the lemmatizer. Since English and other European languages are not highly inflected therefore there have been many stemmers and lemmatizers developed for them, but if we talk about Indian languages, they are highly inflected. Lemmatizer for such languages is rarely found. In this paper we are emphasizing on Hindi language. Hindi is the official language of India. It is most widely spoken in almost all the parts of the country. So, in order to preserve the language and its root words we have developed a lemmatizer which contains various rules. The study however does not include all the rules but can be taken as a prototype for extending the functionality of the system. We have made an attempt to make an automated lemmatizer using the rules. This system can be efficiently used for retrieving the information.

The paper is organized as follows: the first section described the introduction and problems of stemming. The next section discusses the literature work done in this research area. Further we discuss the linguistic background of Hindi. Then we propose our work in which we have shown the methodology and approach that we have used. Further we have shown the processing technique in which we have discussed some examples and provide some limitations of the system depending on the evaluation result. The final section concludes the study of the paper.

2. RELATED WORK

A lot of research work has been done and is still going on for the development of a stemmer as well as lemmatizer. The first stemmer was developed by Julie Beth Lovins [1] in 1968. Later the stemmer was improved by Martin Porter [2] in July, 1980 for English language. The proposed algorithm is one of the most accepted methods for stemming where automatic removal of affixes is done from English words. The algorithm has been implemented as a program in BCPL. Much work has been done in developing the lemmatizer of English and other European languages. In contrast, very little work has been done for the development of lemmatization for Indian languages.

A rule based approach proposed by Plisson et al. [3] is one of the most accepted lemmatizing algorithms. It is based on the word endings where the suffix should be removed or added to get the normalized form. It emphasizes on two word lemmatization algorithm which is based on if-then rules and the ripple down approach. The work proposed by Goyal et al. [6] focuses on the development of a morphological analyzer and generator. They aimed to develop a translation system especially from Hindi to Punjabi. Nikhil K V S [8] built a Hindi derivational analyzer using a specific tool. He used supervised approach by creating a SVM classifier. Jena et al. [9] proposed a morphological analyzer for Oriya language by using the paradigm approach. They classified nouns, adjectives and finite verbs of Oriya by using various paradigm tables. Anand Kumar et al. [7] developed an automatic system for the analysis of Tamil morphology. They used various methodologies, rule based approach and sequence labeling containing the non linear relationships of morphological features from the training data in a better way.

Chachoo et al. [5] used an extract tool named Extract v2.0 for the development of the orthographic component of Kashmiri Script. A method has been proposed by Majumder et al. [12] in which a clustering based approach is used for discovering the equivalent classes of root words. This algorithm was tested for two languages French and Bangla. A rule based approach for stemming in Hindi was proposed by Ramanathan & Rao [11]. The approach is based on stripping off suffixes by generating rules emphasizing on noun, adjective and verb inflections in Hindi. Bharti Akshar et al. [4] proposed the work on natural language processing where they gave a detailed study of morphology using paradigm approach.

3. LINGUISTIC BACKGROUND OF HINDI

Morphemes play a vital role in lemmatization. This is the major way in which morphologists investigate the words. Their formation and internal structure is studied in deep. Morphology is broadly categorized into two parts: derivational morphology and inflectional morphology. Derivational morphology processes the words and form new lexemes from the existing ones. This is done by either adding or deleting affixes. For example – *सजा + वट= सजावट*. Here the word class is changed from adjective to noun. Similarly, in English we have words like *employ + ee = employee*, where the word class is changed from verb to noun. Inflectional morphology processes the words by producing various inflections without changing the word class. For example – *कलम + दान = कलमदान* where both *कलम* and *कलमदान* is noun/singular. The word class remains same here. The root form of the words basically comes under noun and verb classes. This knowledge lead us to trace the paradigm approach. According to Smriti Singh and Vaijayanthi M Sarma [10], Hindi noun classification system shows only the number and case for morphological analysis. Number basically includes either singular or plural. By default a word is kept singular. In Hindi we have two types of Cases – direct and oblique. Oblique words show the case as well as the number of the word. For example – लड़क – ा, लड़क – े, here ा shows singular number whereas े shows plural number. Similarly we have some gender rules. In Hindi, words ending with the suffix ी are termed as feminine whereas the words that end with suffix ा are termed as masculine. For example लड़का, नेता, घोड़ा, कटोरा, बच्चा and many more are masculine ending with ा while लड़की, धोबी,पुत्री, कटोरी, बच्चीand many more are feminine ending with ी. But we have many more words that contradict this concept. For example – the word पानी (water) is masculine, although it is ending with ी. Similarly the word माला (garland) is feminine even though ending with ा. There are some other words from which the suffix cannot be removed. For example – let us consider the suffix ा, the words पिता, माता, बच्चा, कटोरा, नेता, and many more does not require stemming. Such words need to be maintained as it is and should be refrained from being stemmed. So it is found that Hindi is a highly inflected language and needs a deep study of word structure and its formation.

4. PROPOSED WORK

In this paper we have discussed about the creation of a Hindi lemmatizer. The approach for the creation is based on the key concept of *optimization*. Optimization includes both space and time,

so our approach is based on these parameters. The lemmatizer that we discuss here mainly focuses on the time complexity problem. Typically lemmatizer is built using a *rule based approach* and *paradigm approach*. In rule based approach along with the rules, knowledgebase is created for storing the grammatical features. Knowledgebase is also created for storing the exceptional root words. That is we need some root words as it is containing the suffix. Although the knowledgebase creation requires a large amount of memory, but in respect of time it gives us the best, accurate and fast result. The reason behind this fast retrieval is that, a very short time is taken to search the input word from the knowledgebase. The study [7] shows that Tamil words have infinite set of inflections but Hindi words have finite set of inflections which are quite easy to maintain in the knowledgebase. We have restricted our knowledgebase to commonly used words which do not contain the proper nouns like the names of person and place.

4.1 SUFFIX GENERATION

We have gone through various words with their suffixes and examined the morphological changes for the development of a lemmatizer. These suffixes and changes led to the development of specific rules. For example – If we take the word *खराबी* (defect) then we find that the word is derived by adding ी suffix to the word *खराब* (bad) which transform noun to adjective. Similarly there are many other words with the same suffix. Some of them are shown in table 1 and 2.

Table 1. Example of derived words with suffix ी & ई (noun to adjective)

Root Word	Derived Word
साफ़	सफ़ाई
ऊँचा	ऊँचाई
मोटा	मोटाई
गरीब	गरीबी
सर्द	सर्दी

Table 2. Some more suffixes are as follows

Root Word	Derived Word	Suffix
गाड़ी	गाड़ियों	यों
मीठा	मिठाई	ई
पवित्र	पवित्रता	ता
जादू	जादूगर	गर
रोशन	रोशनदान	दान
चढ़	चढ़ाई	ाई

Hindi words are large in number and for this reason the extraction of suffixes had a large list. Since the work has been done manually therefore this phase was quite time consuming. The

suffixes were generated by processing a corpus of 40,000 sentences from which 75 lakh words were manually stemmed among them 124 suffixes were derived.

4.2 RULES GENERATION

After generating the suffix list we have developed rules. We have created 124 rules which are framed in such a way that the suffix gets removed from the input word and if required, addition of character or 'maatra' takes place. For example – let us take the suffix ोों. Some of the words containing this suffix are shown in table 3.

Table 3. Words showing the suffix ोों

		Rule application	
Word	Root	Extraction of suffix	Addition of character
लड़कों	लड़का	ोों	ा
सड़कों	सड़क	ोों	—
लेखकों	लेखक	ोों	—

In the above table on removing the suffix ोों we get their respective root word, but the word लड़कों is an exception here because on removing the suffix ोों we need to add ा to the last letter of the word to make it a genuine root word' लड़का.' Similarly there are many other rules for removing the suffix and if necessary addition of character may also takes place. Similarly we also have some other rules, like the rule for extracting the suffix ियों which is shown in table 4.

Hindi has a grammar rule, according to which when the plural is removed, we need to add ी to the last letter of the word. Table 4 mentions the rule for the suffix ियों in which we have created a general rule for removing the suffix and adding ी to the word, but we have some exceptions here which include the addition of ि instead of ी. In the below table we have also shown an exception in the last word चिड़ियों where the root form is चिड़िया. The word चिड़ियों contains two suffixes together which are ियों and ोों. This becomes hard for the system as it finds difficulty in picking up the correct rule for the particular word. Similarly there are many more exceptions for which we have generated different rules. To overcome such problems we have built a database in which such exceptional words are kept. Although this work requires much time but for the sake of fast and accurate result this approach is applied.

Table 4. Words showing the suffix ियों

		Rule application	
Word	**Root**	**Extraction of suffix**	**Addition of character**
लड़कियों	लड़की	ियों	ी
कहानियों	कहानी	ियों	ी
कवियों	कवि	ियों	ि (exception)
चिड़ियों	चिड़िया	ियों	िया (exception)

5. PROCESSING TECHNIQUE

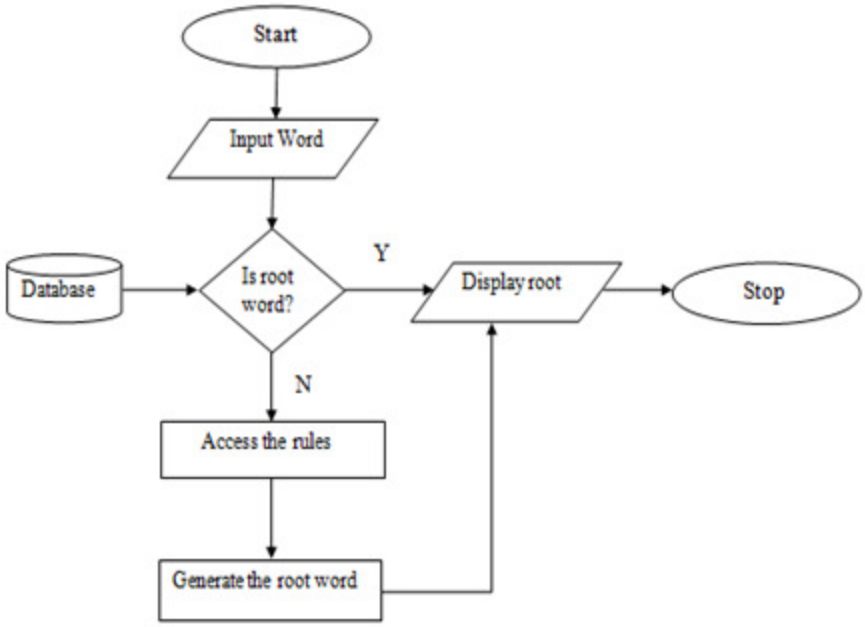

Fig 1. Schematic diagram of the system

The foremost step is to read the input word. The database contains all the root words. The input is checked in the database, if it is present in the database then the word is displayed as it is. If the word is not present in the database then it comes down to access the rules. After accessing the rules the root word is generated and displayed. The rule is followed as -

```
If (root) present in (root list)
{
Fetch the root from the list
Display;
}
else if (root) not present in (root list)
{
If (source) ends with (suffix)
{
```

```
Substring the source
Display the root;
}
}
```

6. ILLUSTRATION

As an illustration we gave a set of Hindi words in order to analyze the output. Some of them are –

Table 5. Stemmed Output

Input	Output
चिड़ियों	चिड़िया
लड़कियों	लड़की
भारतीयता	भारत
प्रतिभाशाली	प्रतिभा
गौरवांवित	गौरव

7. EVALUATION

The system is evaluated for its accuracy where we gave 2500 words for analysis. Among these 2500 words 2227 words were evaluated correctly and 273 words were incorrect because they violated both the exceptional and general rules.

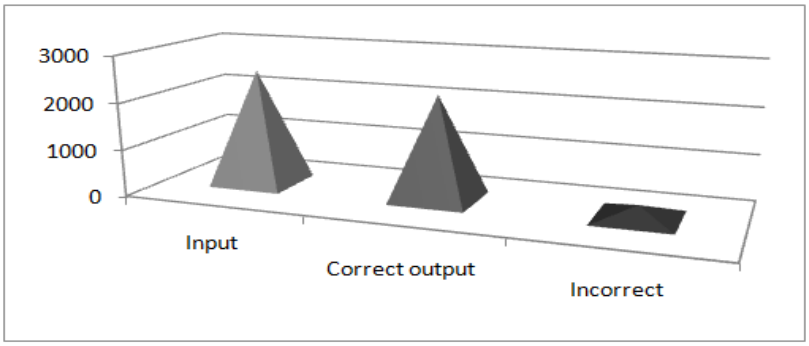

Fig. 1. Graphical representation of evaluation

Our system gave 89.08% accuracy. We used the following formula to calculate the accuracy.

Accuracy % = Total correct lemmas / Total words

8. CONCLUSION

In this paper we have discussed the development of Hindi lemmatizer. The work focuses on rule based approach along with this paradigm approach is also used in which we have created knowledgebase containing all the Hindi root words that are commonly used in day to day life. The main aim is emphasized on time optimization problem rather than on space. Since nowadays

space is not at all a big problem, therefore our approach aimed to optimize time and generate accurate result in a very short period. Our system gave 89.08% of accuracy.

REFERENCES

[1] Julie Beth Lovins, Development of stemming Algorithm, Mechanical Translation and Computational Linguistics, Vol. 11, No. 1, pp 22-23, 1968.
[2] Martin F. Porter, An algorithm for suffix stripping, Program, Vol. 14, No. 3, pp 130-137,1980.
[3] Plisson, J, Larc, N, Mladenic, D.: A Rule based approach to word lemmatization, Proceedings of the 7th International Multiconference Information Society, IS-2004, Institut Jozef Stefan, Ljubljana, pp. 83-86, 2008.
[4] Bharti Akshar, Vineet Chaitanya, Rajeev Sangal, Natural Language Processing: A Paninian Perspective. Prentice-Hall of India, 1995.
[5] Manzoor Ahmad Chachoo, S.M.K. Quadri. Morphological Analysis from the Raw Kashmiri Corpus Using Open Source Extract Tool, Vol. 7, No. 2, 2011.
[6] Vishal Goyal, Gurpreet Singh Lehal, Hindi Morphological Analyzer and Generator, IEEE Computer Society Press, California, USA pp. 1156-1159, 2008.
[7] Anand Kumar M, Dhanalakshmi V, Sonam K P, A sequence labeling approach to morphological analyzer for tamil language, International Journal on Computer Science and Engineering, Vol. 20, No. 06, 2010.
[8] Nikhil K V S, Hindi derivational morphological analyzer, Language Technologies Research Center, IIIT Hyderabad, 2012.
[9] Itisree Jena, Sriram Chaudhary, Himani Chaudhary, Dipti M. Sharma, Developing Oriya Morphological Analyzer Using Lt-toolbox, ICISIL 2011, CCIS 139, pp. 124-129, 2011.
[10] Smriti Singh, Vaijayanthi M Sarma. Hindi Noun Inflection and Distributed Morphology.
[11] A. Ramanathan and D.D Rao, A Light Weight Stemmer for Hindi, In Proceedings of Workshop on Computational Linguistics for South Asian Languages, 10th Conference of the European Chapter of Association of Computational Linguistics, pp. 42-48, 2003.
[12] Prasenjit Majumder, Mandar Mitra, Swapan k. Pauri, Gobinda Kole, Pabitra Mitra and Kalyankumar Datta, YASS: Yet Another Suffix Stripper, ACM Transactions on Information Systems, Vol.25, NO.4, pp. 18-38, 2007.