International Journal of Research in Advent Technology (E-ISSN: 2321-9637) Special Issue
1st International Conference on Advent Trends in Engineering, Science and Technology
"ICATEST 2015", 08 March 2015

Survey paper of Different Lemmatization Approaches

Riddhi Dave[1], Prem Balani[2]
[1]*ME Student, Information Technology Department, GCET, GTU affiliated, V.V. Nagar, Gujarat, India,* riddhidave1309@gmail.com
[2]*Assistant Professor, Information Technology Department, GCET, GTU affiliated, V.V. Nagar, Gujarat, India,* prembalani@gcet.ac.in

Abstract: Lemmatization is use to normalize inflectional form of word to its root word. So it can be used as pre-processing step in any natural language processing application. Lemmatization is very important approach for information retrieval process. Lemmatization is used to reduce different inflectional form as well as derivational form of word to its root or head word which called as its 'lemma'. A 'lemma' is the simply the "Dictionary form" of a word. In lemmatization, different grammatical form of word can be analyzed as a single word. In this paper we have discussed five different Lemmatization approaches. The first one is Edit Distance on dictionary algorithm which is combination of string matching and most frequent inflectional suffixes model. Second is Morphological Analyzer which is based on "finite state automata". Third approach uses "radix trie" data structure which allow retrieving possible lemma of a given inflected or derivational form. Fourth approach is Affix lemmatizer which is combination of rule based and supervised training approach and last approach is fixed length truncation approach.
Keywords – Lemmatization, Information Retrieval,

1. INTRODUCTION

As the language is an important tool for communication, so natural language processing is concerned with the interaction between human languages and computers. NLP involves enabling computers to derive meaning from human or natural language input. Natural language processing is very hot research topic now a days, as it is used in most of the linguistic activities.

An information retrieval process is the major activity in natural language processing. Information retrieval is the process of obtaining resources as per need from the avble resources.

An information retrieval process begins when a user enters a query into the system. Queries are formal statements of information needs, for example search strings in web search engines. In information retrieval a query does not uniquely identify a single object in the collection. Instead, several objects may match the query, perhaps with different degrees of relevancy.[4] "Lemmatization" refers to normalized different inflectional forms as well as derivational forms to its head word.

This task can be used as a pre-processing step for many natural processing applications (e.g. morphological analyzers, electronic dictionaries, spell-checkers, stemmers, etc.). It may also be useful as a generic keywords generator for search engines and other data mining, clustering and classification tools.[1]

Normalization is very important task in any natural language processing application. Stemming or Lemmatization used as a normalized technique to reduce different grammatical words to its head word by applying set of rule. Both stemming and lemmatizing can be used as a pre processing steps in IR application.

Stemming is process of reducing different inflectional form to its stem by applying different set of rule. Aim of Stemming is just to reduce word to its stem without bothering about POS. It is used in most of the text mining application where aim is just to reduce the form of word without worrying of its occurrence in the given context. So it is used to convert the different inflectional form of word to its stem. The result of stemming is called as a stem, it is not always a dictionary word.

In linguistics, a lemma (from the Greek noun "lemma", "headword") is the "dictionary" or "canonical" form of a set of words. More specifically, a lemma is the canonical form of a lexeme, where lexeme refers to the set of all the forms that have the same meaning, and lemma refers to the particular form that is chosen as base form to represent the lexeme.[2] Lemmatization used as a most frequently used normalization technique in any information retrieval application like indexing and searching.

Lemmatization aims to remove inflectional endings only and to return dictionary form of a word and may use of a vocabulary and/or morphological analysis of words. Therefore lemmatizers require much

knowledge about language than stemmers and they don't use language specific rules unlike stemmers. Lemmatization is closely related to stemming, however, stemming operates only on a single word at a time. Instead, lemmatization may operate on the full-text and therefore can discriminate between words that have different meanings depending on part-of-speech. On the other hand, stemmers are typically easier to implement and run faster. Hence, lemmatizers play a significant role in IR and ability to lemmatize words efficiently and effectively is thus important.[2]

In this paper we discussed five different approaches of lemmatization. First approach is Edit distance on dictionary based approach. It is combination of string matching and most frequent inflectional suffix model. String matching is performed between the dictionary word and word given in to the query string. Second is Morphological Analyzer which is based on finite state automata. Third approach is radix trie approach. It is also known as tree approach so search for given query string can be done from top to bottom. Fourth is Affix lemmatizer. It is rule based approach where set of rules are defined based on language knowledge. By using the defined set of rules affixes is removed from the inflectional and derivational words and produce lemma. In additional to affix removal it used training of data. This makes it more accurate. This approach is the fastest approach among all and fifth approach is Fixed length truncation approach where fixed size of suffix removed from the given word and rest is returned as a result.

The rest of the paper is organized as bellow. The next section 2, explains different approaches of lemmatization. Section 3, conclude the paper and Section 4 contains future enhancement.

2. APPROACHES OF LEMMATIZER

We have study five lemmatization approaches. First approach is string matching dictionary based approach. Second is based on finite state automata. Third approach is based on trie approach, it is also known as tree approach. Trie approach retrieve all possible lemma of a given word inflectional words. Fourth approach is affix removal approach and last one is fixed length truncation approach. Last approach mostly used for those language where size of word is more than 7. So by removing fixed size of suffix it can produce good result.

a) Levenshtein Distance Dictionary based Approach

The Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other.[5]
Searching similar sequences of data is of great importance to many applications such as the gene similarity determination, speech recognition applications, database and/or Internet search engines, handwriting recognition, spell-checkers and other biology, genomics and text processing applications. Therefore, algorithms that can efficiently manipulate sequences of data (in terms of time and/or space) are highly desirable, even with modest approximation guarantees.[1]

The Levenshtein Distance of two strings A and B is the minimum number of character transformation required to convert string A to string B.

The following Equation 1 is used two find the Levenshtein distance between two strings a, b is given by $lev_{a,b}(|a|,|b|)$ where

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & if\ min(i,j) = 0, \\ min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & otherwise \end{cases}$$

Equation 1: Levenshtein Distance between two strings

Where $1(a_i \neq b_j)$ is indicator function equal to 0 when $a_i = b_j$ and equal to 1 otherwise.

Note that the first element in the minimum corresponds to deletion (from a to b), the second to insertion and the third to match or mismatch, depending on whether the respective symbols are the same.[5]

The edit distance algorithm is performed by using three most "primitive edit operation". By term primitive edit operation we refer to the substitution of one character to another, the deletion of a character and insertion of a character. So this algorithm can be performed by three basic operations like insertion, deletion and substitution.

Some approached focused on suffix phenomena only. But this approach deals with both suffixes as well as prefixes. So it is known as affixation phenomena.

Sometime it happens that suffixes added into the words based on grammatical rules. For example word "going", this approach return headword "go". But for word "went", it contains discrete entry of lemma in dictionary.

The idea is to find out all possible lemma for user's input word. It contain a file which is having 30,000 possible lemmas stored.

For each one of the target words, the similarity distance between the source and the target word is calculated and stored. When this process is completed, the algorithm returns a set of target words having the minimum edit distance from the source word.[1]
So algorithm compare user input to the all available stored lemmas. Retrieve the minimum distance word from the target word .

The algorithm provides the option to select the value of the approximation that the system considers as desired similarity distance (e.g. if the user enters zero as the desired approximation, then only the target words with the minimum edit distance will be returned, whereas if he/she enters e.g. 2 as the desired approximation, then the returned set will contain all the target words having a distance <=(minimum + 2) from the source word.[1]

This approach also distinguishes words like "entertained" and "entertainment". Its return entertain for the entertained word but not for entertainment, because entertainment its self is a noun and its different than entertained.

b) Morphological Analyzer based Approach

Morphological Analyzer gives all possible analyses for a given word which is based on finite state technology, and it produces the morphological analysis of the word form as its output.[2]

This approach uses finite state automata and two level morphology to build a lexicon for a language with infinite vocabulary.

Two-Level rules are declarative constraints that describe morphological alternations, such as the y->ie alternation in the plural of some English nouns (spy->spies). [6]

Aim of this approach is to converts two-level rules into deterministic, minimized finite-state transducers. It describes the format of two-level grammars, the rule formalism, and the user interface to the compiler. It also explains how the compiler can assist the user in the development of a two-level grammar.[6]

A finite state transducer (FST) is a finite state machine with two tapes: an input tape and an output tape. This contrasts with an ordinary finite state automaton (or finite state acceptor), which has a single tape.[7]

Transducer means to translate a word from one state to another. Transducer is having two state, one is input tape and another is output tape.

Finite state transducer is 6-tuple (Q, Σ, Γ, I, F, δ) such that:

- Q is a finite set, the set of states;
- Σ is a finite set, called the input alphabet;
- Γ is a finite set, called the output alphabet;
- I is a subset of Q, the set of initial states;
- F is a subset of Q, the set of final states; and

$$\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \times Q$$

(where ε is the empty string) is the transition relation.[7]
FSM give input actions and output depends on only state. State change from input tap to output tap based on this action performed.

For example entry for the action is "open" starts a motor opening the door, the entry action in state "Closing" starts a motor in the other direction closing the door. States "Opened" and "Closed" stop the motor when fully opened or closed. They signal to the outside world (e.g., to other state machines) the situation: "door is open" or "door is closed".[7]

So FSM takes action as input which can be any rule or operation and generate output tap from current input tap. So this approach mostly used for computational morphology and phonology.

c) *Radix Trie based Approach*

In computer science, a radix tree (also patricia trie or radix trie or compact prefix tree) is a space-optimized trie data structure where each node with only one child is merged with its parent. This makes them much more efficient for small sets (especially if the strings are long) and for sets of strings that share long prefixes.[9]

Trie is a data structure which allows to retrieve all possible lemmas. Here each node is having single character. Two nodes connected with the edges. Word is retrieve byte by byte. This approach is also involve backtracking for getting appropriate result.

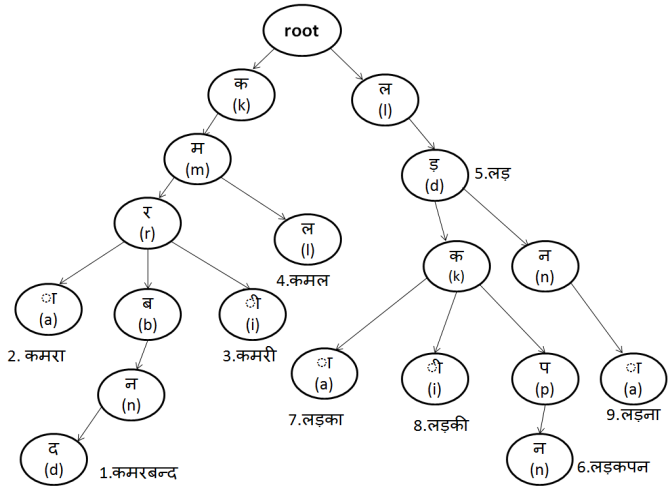

Figure 1: A simple trie storing Hindi words[8]

Word is stored in root, in character by character unicode byte order. User input word searched from first node and it traverse the tree up to last character of the word. It is possible that traverse need to backtrack for some level.

Lemmatizer gives the following output:

(ल लड़ लड़का लड़की लड़कपन लड़कोरी लड़कौरी)

Figure 2: Example for search "Ladkiyan" Hindi word[8]

So search word up to third word "Ladka" as shown in Figure 2. But after that it needs to performed backtrack for one level reach to the word "Ladk" and then traverse again and get "Ladki" the correct word as shown in Figure 1.

So just to use radix tree approach cannot give accurate result. But performing backtracking for one or two level it gives most accurate result.

d) Affix Lemmatizer

The most common approach for word normalization is to remove affix from a given word. Suffix or prefix removed as per rules defined based on grammatical knowledge of the language. To just remove suffix or prefix from word cannot give accurate head word or root word.

To just used rule based approach cannot give accurate result so by combining rule based approach to some statistical approach like supervised training can give more accurate result.

Supervised training algorithm generates a data structure consisting of rules that a lemmatizer must traverse to arrive at a rule that is elected to fire. [10]

After training, the data structure of rules is made permanent and can be consulted by a lemmatizer. The lemmatizer must elect and fire rules in the same way as the training algorithm, so that all words from the training set are lemmatized correctly. It may however fail to produce the correct lemmas for words that were not in the training set – the OOV words.[10]

For training word this approach used prime and derived rules. Prime rule for training is the least specific rule needs to lemmatize. Where derived rules are more specific rule-can be created by adding or removing characters.

For example rule can be "watcha" which is derived from what are you, "yer" which is derived from you are rather than "your".

This approach is more generalized than only suffix removal approach.

The bulk of 'normal' training words must be bigger for the new affix based lemmatizer than for the suffix lemmatizer. This is because the new algorithm generates immense numbers of candidate rules with only marginal differences in accuracy, requiring many examples to find the best candidate.[10]

e) Fixed length truncation

In this approach, we simply truncate the words and use the first 5 and 7 characters of each word as its lemma. In this approach words with less than n characters are used as a lemma with no truncation.[2] This approach is most appropriate for the languages like Turkish which has average length of word is 7.07 letters.

So this approach is used when time is most priority issue. It is the simplest approach not dependent on any language or grammar. So it can be applicable to any language.

3. CONCLUSION

As we have discussed that only rule based approach can give the root word. It is not always an efficient solution because space needed for storing the predefined rules is big issue. So by combing rule based to some statistical approach can give more accurate result.

To use language independent approach is efficient solution. By the term "language-independent", we mean that the algorithm can perform sufficiently well for a variety of languages regardless of the specific grammar and inflectional rules that apply to them. So for language independent approach Levenshtein edit distance is best solution.

Another solution is to use some data structure like radix tree can be optimal solution. It is the longest prefix match functionality, which is able to find most appropriate lemma of the input word.

4. FUTURE ENHANCEMENT

Although research has been done in developing lemmatizer, still there are statistical approach or data structure available which are used for linguistic purpose. By using it we can achieve best lemmatizer which is most save both time as well as space.

REFERENCES

[1].Dimitrios P. Lyras, Kyriakos N. Sgarbas, Nikolaos D. Fakotakis, "Using the Levenshtein Edit Distance for Automatic Lemmatization: A Case Study for Modern Greek and English," Tools with Artificial Intelligence, 19th IEEE International Conference , pp.429-435, 29-31 October, 2007.

[2].Okan Ozturkmenoglu, Adil Alpkocak, "Comparison of Different Lemmatization Approaches for Information Retrieval on Turkish

Text Collection," Innovations in Intelligent Systems and Applications (INISTA), 2012 IEEE International Symposium, ,pp.1-5, July 2012.

[3].Snigdha Paul, Nisheeth Joshi, Iti Mathur, "Development of a Hindi Lemmatizer," International Journal of Computational Linguistics and Natural Language Processing (IJCLNLP), pp.380-384, vol.2, May 2013.

[4].http://en.wikipedia.org/wiki/Information_retrieval

[5].http://en.wikipedia.org/wiki/Levenshtein_distance

[6].L. Karttunen, K. R. Beesley, "Two-level rule compiler," Palo Alto,XEROX: Research Center-Technical Report, 1992.

[7].http://en.wikipedia.org/wiki/Finite_state_transducer

[8].Pushpak Bhattacharyya, Ankit Bahuguna, Lavita Talukdar and Bornali Phukan, "Facilitating Multi-Lingual Sense Annotation: Human Mediated Lemmatizer", Global Wordnet Conference (GWC 2014), Tartu, Estonia, 25-29 January, 2014

[9].http://en.wikipedia.org/wiki/Radix_tree

[10]. Bart Jongejan, Hercules Dalianis, "Automatic training of lemmatization rules that handle morphological changes in pre-, in- and suffixes alike", 47th Annual Meeting of the Association for computational linguistics (ACL) and the 4th International Joint Conference on Natural Language Processing (IJCNLP) of the AFNLP, p.p.145-153, August 2009.