Stemming Techniques for Tamil Language

Vairaprakash Gurusamy
Department of Computer Applications, School of IT,
Madurai Kamaraj University, Madurai
vairaprakashmca@gmail.com

K.Nandhini
Technical Support Engineer, Concentrix India Pvt Ltd, Chennai,
Nandhinii.k92@gmail.com

***Abstract*-- Preprocessing is an important task and critical step in Text mining, Natural Language Processing (NLP) and information retrieval (IR). In the area of Text Mining data Preprocessing used for extracting interesting and non-trivial and knowledge from unstructured text data. Information Retrieval (IR) is essentially a matter of deciding which documents in a collection should be retrieved to satisfy a user's need for information. The user's need for information is represented by a query or profile, and contains one or more search terms, plus some additional information such as weight of the words. Hence, the retrieval decision is made by comparing the terms of the query with the index terms (important words or phrases) appearing in the document itself. The decision may be binary (retrieve/reject), or it may involve estimating the degree of relevance that the document has to query. Unfortunately, the words that appear in documents and in queries often have many morphological variants. So before the information retrieval from the documents the data preprocessing techniques are applied on the target data set to reduce the size of the data set which will increase the effectiveness of IR System. Stemming is one of the most important preprocessing technique which reduces all the words into their root word by stripping both prefixes and suffixes. In this paper, we discuss the various Tamil Stemming algorithms and the issues about the each algorithm**

Keywords-- Preprocessing, Stemming, Light Stemmer, Affix stripping Stemming

I. Introduction

Stemming is the process of conflating the variant forms of a word into a common representation, the stem. For example, the words: “presentation”, “presented”, “presenting” could all be reduced to a common representation “present”. This is a widely used procedure in text processing for information retrieval (IR) based on the assumption that posing a query with the term presenting implies an interest in documents containing the words presentation and presented

Errors in Stemming

There are mainly two errors in stemming.

1. Over stemming
2. Under stemming

Over-stemming is when two words with different stems are stemmed to the same root. This is also known as a false positive.

Under-stemming is when two words that should be stemmed to the same root are not. This is also known as a false negative. Light-stemmers reduces the over-stemming errors but increases the under-stemming errors. On the other hand, heavy stemmers reduce the under-stemming errors while increasing the over-stemming errors

Light Stemmer

Light stemmer is one type of rule based stemmer. It works by truncating all possible suffixes form and produce finite verb. Light stemming is used to find the representative indexing form of given word by the application of truncation of suffixes. The core objective of light stemmer is to preserve the word meaning intact and so increases the retrieval performance of an IR system. A Light Stemmer Algorithm for the Tamil Language is projected in Figure1

Light stemmer Algorithm for Tamil Language
Input : List of Tamil words Output : Stemmed(Root) words
Step 1 : Eliminate the entire complex plural. Step 2 : After the plural word is converted into singular word, during the iteration, the word is also checked for adjective; if it is found, then its equivalent verb is substituted. Example, the term 'diya(ÊÂ)' in Odiya(µÊÂ) will be changed to 'du(Î)' and the word is changed to 'Oodu(µÎ)'. Step 3 : After the adjectives are converted to main word, the tenses are eliminated such that Paadiya(À¡ÊÂ), Paadukinra(À¡Î¸ ¢ýÈ) and Paadum(À¡Îõ) will be changed to Paadu(À¡Î). Step 4 : According to the identified suffix, the next possible suffix list is generated. Step 5 : The Light algorithms are used for plural to singular conversion, and for adjective and tense words to main verbs conversion.

Fig. 1 Light Stemmer Algorithm for the Tamil Language

Improved Light Stemmer

Light stemmer works by truncating all possible suffixes form and produce finite verb. Light stemming is used to find the representative indexing form of given word by the application of truncation of suffixes. The core objective of light stemmer is to preserve the word meaning intact and so that it increases the retrieval performance of an IR system. The proposed improved light stemming algorithm in this system defines an algorithm (removes suffixes recursively and a single prefix non-recursively), that is called SP. The same defined affixation terms list were used but with a modified execution step via Suffix Prefix Suffix truncating process, that algorithm is called SPS.

Notice that most of Tamil words use (Thiru, Thirumathi) prefix as a declarative term (e.g., Thiru. Dr.A.P.J.AbdulKalam) therefore, proposed two new major categories in classifying of the designed algorithms; Without-Thiru (WOTH the stemmer accepts the non-stemmed words after removing the prefixed Thiru) and With Thiru (WTH stemmer acquires the whole non-stemmed word). This work applies improved light stemming concept to replace plural terms, adjectives and tense words. The flowchart and algorithm of improved light stemmer projected in Fig.2 and Fig.3.

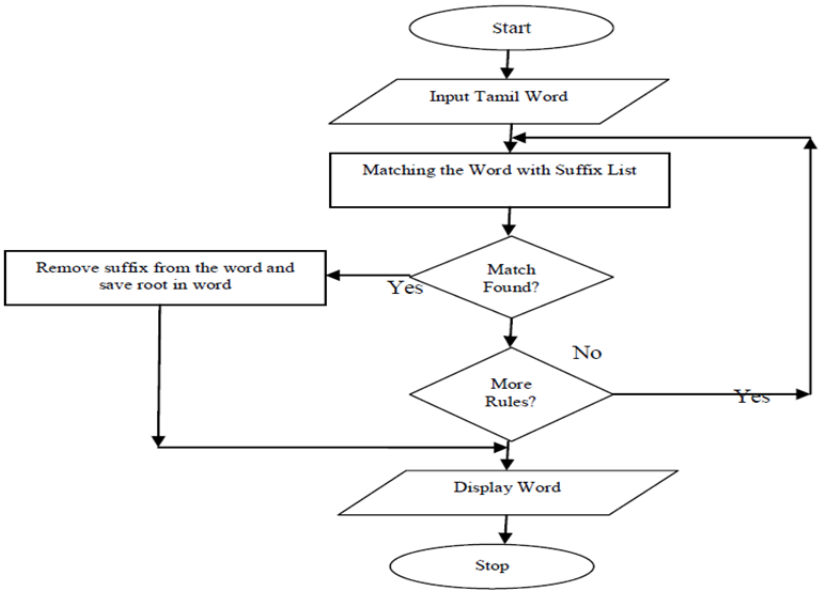

Fig. 2 Flow chart for Improved Light Stemmer Algorithm for Tamil Language

Improved Light stemmer Algorithm for Tamil Language

Input: List of preprocessed Tamil words
Output: Stemmed(Root) words

Step 1: Eliminate the entire complex plural.

Step 2: After the plural word is converted into singular word, during the iteration, the word is also checked for adjective; if it is found, then its equivalent verb is substituted. Example, the term 'diya(ÊÂ)' in Odiya(µÊÂ) will be changed to 'du(Î)' and the word is changed to 'Oodu(µÎ)'.

Step 3: After the adjectives are converted to main word, the tenses are eliminated such that Paadiya(À¡ÊÂ), Paadukinra(À¡Î¸ýÈ) and Paadum(À¡Îõ) will be changed to Paadu(À¡Î).
Step 4: According to the identified suffix, the next possible suffix list is generated and add more rules.
Step 5: The Light algorithms are used for plural to singular conversion, and for adjective and tense words to main verbs conversion.

Figure3: Improved Light Stemmer Algorithm for the Tamil Language

II. Proposed Algorithm

In Tamil, suffixes are used for many things like tense, plurality, person etc. So the suffixes are grouped into categories and a routine is defined for each category to handle the removal the respective suffixes. After removal of suffix for each category there is routine to fix or recode the ending of the word to make it consumable for the next routine. Also before stripping the suffix every routine check for the current size of the string.

As shown in Figure 4, first the prefixes are removed followed by the suffixes. Every suffix stripping routine checks for the length of the string before proceeding and after removing a suffix calls the routine responsible for fixing the endings.

WHY AN AFFIX STRIPPING ALGORITHM?

An affix stripping algorithm was chosen for the following reason:

1. An affix stripping algorithm does not require a dictionary. In Tamil, the suffixes are attached in an order. So a stemming algorithm which stems most of the words satisfactorily

can be designed without the help of a dictionary.

2. The algorithm is very fast. The algorithm needs not lookup any dictionary or does complex statistical analysis based on any collected corpus. It just works on the string to be stemmed. So it is very fast.

3. Since it does not require any supporting data the algorithm can be run on any device. For example, to port any dictionary based stemmer to a low memory device the dictionary might need to be trimmed down thereby reducing the accuracy of the stemmer. But an affix stripping algorithm does not have any such memory requirements and does not hold much data during its operation

4. There is a lack of quality corpus to train statistical algorithms.

The Flowchart for Rule Based Iterative Affix Stripping Algorithm for Tamil Language is given below

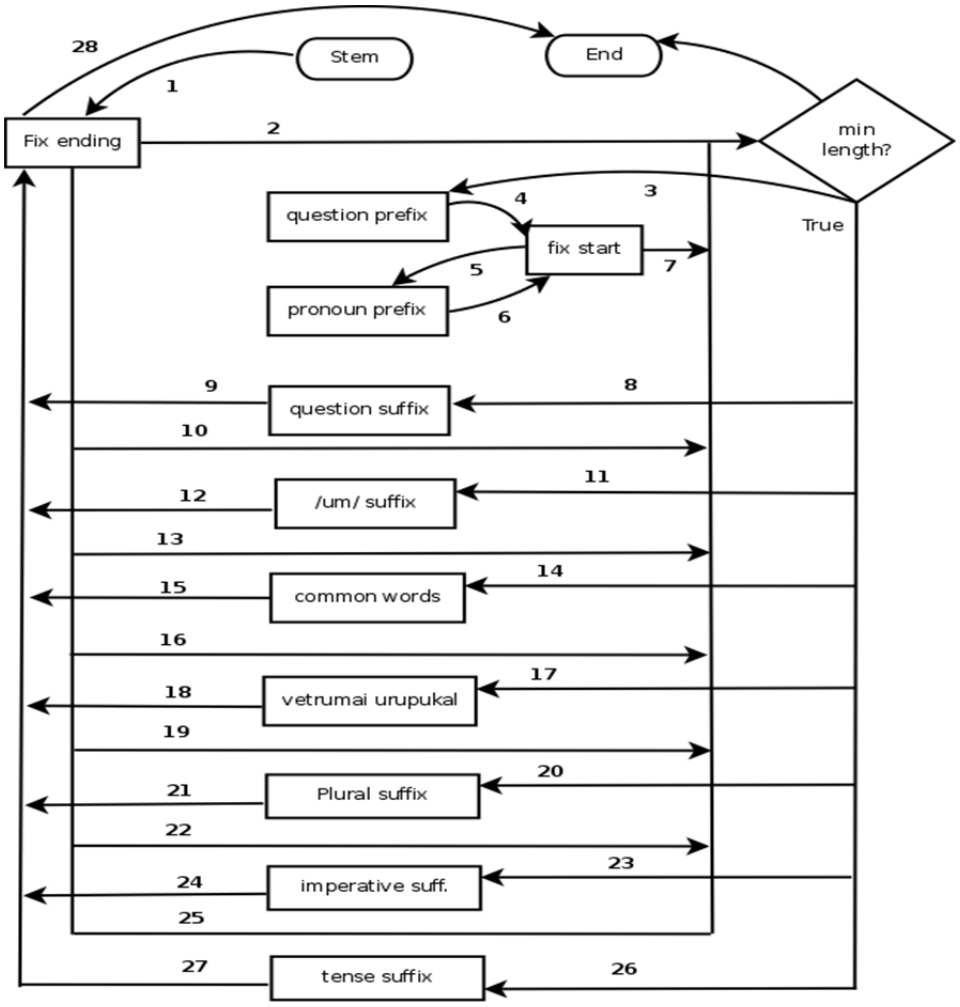

Fig. 4 Flowchart for Rule Based Iterative Affix Stripping Algorithm for the Tamil Language

Minimum length criteria

Being a strong stemmer it has a tendency to over stem some words to single letters. To prevent this every routine check for the length of the string. Currently, the minimum length is set as 4 characters. These are not 4 characters exactly since in Unicode a meaningful character can be represented by more than one code points. So the check made in the implementation actually only verifies the number of code points in the string than the actual meaningful characters. Also, the routine which fixes the ending does not check for the length of the string. So it is still possible to get a stem of length one character.

Prefix Removal:

There are two routines in the algorithm to handle prefixes. One is for handling the prefix in the questions. Eg. *எக்காலம்* (which period?). - *காலம்*. Another one is for removing the pronoun prefixes, *அ*, *இ* and *உ*.Eg. *அக்காலம்*(that period).- *காலம்*

After removing the prefixes another routine handles fixing the start of the word. The above prefixes introduce *வ்* when the root word starts with a vowel. *வ* in the start of the word cannot combine with certain vowels. In such cases this routine substitutes with appropriate vowel as the starting.

Fixing the ending:

When a suffix joins a root word one of the followings can happen

1. New letters are introduced
2. Some letters are removed
3. The letters are transformed
4. Joins naturally without addition/removal

Fix_ending routine tries to handle these modifications before the next suffix removal routine is called.

If the join had caused new letters to be introduced, this routine removes it. For example, vallinam consonants appear as conjunctions in many cases. A normal word will not end with a vallinam consonant.

கடக்க	கடக்	கட
original word	suffix stripped	vallinam consonant removed

If the join has caused some characters to be removed it leaves it since it is possible for more than one valid character to be appropriate candidates.
If the join has transformed some of the characters it tries to recode it. It currently cannot recover all such transformations.

Example:

மரத்தின்	மரத்த்	மரம்
original word	suffix removed	end recoded

Suffix removal:

The stemming algorithm handles different kinds of suffixes. They are discussed in the following sections

1) Question suffixes:

This routine removes the suffixes. The suffixes are ஆ, ஏ, ஓ.

கண்ணனா	கண்ணன்
Is it Kannan	Kannan

2) Conjunction suffix:

This routine removed the suffix உம்

அவனும்	அவன்
Him and	Him

3) Common words:

This algorithm tries to remove some of the common words that are attached to verbs or nouns. These are not suffixes and are proper words.

அவனில்லாத	அவன்
without him	Him

4) Case suffixes:

Tamil case suffixes are attached to the ends of nouns to express grammatical relations (e.g., subject, direct object, etc.) as well as meanings typically expressed in English through prepositions (e.g., 'in', 'to', 'for', 'from', etc.).

அவனிடம் (with him)	அவன் (him)
மரத்தில் (in tree)	மரம் (tree)

5) Plural suffix:

The plural suffix in Tamil is கள்.

மரங்கள்	மரம்
Trees	Tree

6) Imperative suffixes:

These are used to command a person.

காண்பி (show me)	காண் (see)

7) Tense suffixes:

This routine removes tense indicating suffixes. It also includes person suffixes.

பிரிகின்றன (leaving)	பிரி (leave)

Apart from the standard suffixes the routine also removed கொண்டு and similar words.

III. Algorithm Implementation

Snowball

he Rule Based Iterative Affix Stripping algorithm for the Tamil language described in the previous chapter was implemented using Snowball language. Snowball is a small string handling language mainly designed to define stemming algorithms in a natural way. The language was created by Dr. Martin Porter when he saw various buggy implementations of his famous Porter algorithm for English. The reasons for errors in the implementation can be grouped into following: misunderstanding of the original algorithm, errors in handling the encoding and the programmers urge to improve the algorithm. The language was mainly developed to avoid such implementation errors and it widely used now for developing stemming algorithms. Stemming algorithms for many languages like German, French, and Turkish etc have been implemented using the Snowball language

Challenges and Limitations

The Rule Based Iterative Affix Stripping algorithm for Tamil language having the following limitations,

1. It can't handle irregular forms
2. It can't handle compound words
3. It susceptible Over/Under-Stemming
4. Stems are not valid in more cases
5. Diglossia of language poses more challenges

IV. Conclusion

Due to a lot of limitations in the Rule Based Iterative Affix Stripping algorithm for Tamil language, still, it having more space and complex to develop the new algorithm which eliminates the issues that discussed above.

Reference

[1] Vairaprakash Gurusamy, S Kannan, "Preprocessing Techniques for Text Mining" https://www.researchgate.net/publication/273127322
[2] M.A.Nuhman (1999), Basic Tamil Grammar, University of Peradeniya, Sri Lanka.
[3] K.Rajan ,Dr. V.Ramalingam, Dr.M.Ganesan."Machine Learning of sandhi Rules for Tamil"
[4] Rajendran, S., Arulmozi, S., Ramesh Kumar, Viswanathan, S. 2001. "Computational morphology of verbal complex," Paper read in Conference at Dravidan University, Kuppam, December 26-29, 2001.
[5] Ramachandran, Vivek Anandan and Krishnamurthi, Ilango. "An Iterative Suffix Stripping Tamil Stemmer". Proceedings of the International Conference on Information Systems Design and Intelligent Applications (2012): Volume 132, 583-590
[6] Ms. Anjali Ganesh Jivani "A Comparative Study of Stemming Algorithms" Int. J. Comp.Tech. Appl., Vol 2 (6), 1930-1938
[7] M.Thangarasu and Dr.R.Manavalan, Design and Development of Stemmer for Tamil Language: International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 7, July 2013.
[8] Porter M.F. "Snowball: A language for stemming algorithms". 2001.
[9] http://snowball.tartarus.org/texts/introduction.html
[10] Porter M, An algorithm for suffix stripping Program, 143, Pp. 130-137, 1980