A Rule Based Iterative Affix Stripping Stemming Algorithm For Tamil

Damodharan Rajalingam

ABSTRACT

Stemming is an important step in many of the Information Retrieval (IR) and Natural Language Processing (NLP) tasks. Stemming reduces derived forms of words to a common root. When used in IR it increases the recall performance. Stemming algorithms are very specific to a languages as different languages have different rules for derivation. For a language to have better IR and NLP tools stemming algorithm is a basic necessity. There is currently no open implementation of a stemming algorithm available for Tamil. There might be proprietary products that include a stemming algorithm for their uses but having an openly available version will help in implementation of IR and NLP tools for Tamil.

This paper discusses about the implementation of a stemming algorithm that is available[1] as Open Source Software. The algorithm implemented is a rule based iterative affix stripping algorithm. The algorithm is implemented using Snowball[2], a string processing language specifically used for implementing stemming algorithms. The algorithm was tested against the Tamil WordNet data. The results of these tests are also presented in this paper.

Categories and Subject Descriptors

H.3.1 [**Content Analysis and Indexing**]: Linguistic processing, I.2.7 [**Natural Language Processing**]: Text analysis

General Terms

Algorithms, Design, Languages

Keywords

Stemming

1 INTRODUCTION

The problem of information storage and retrieval has been receiving more and more attention in the recent years. A more obvious example is how much people depend on web search engines like Google, Yahoo etc in their everyday life. With the growth of the World Wide Web the amount of information being generated is growing at a tremendous rate. This has created a need for faster and better information retrieval systems. The requirement for a good retrieval system is not limited to internet only. Users are having a lots of data in their personal computers that the old method of maintaining hierarchy of folders is not a viable way of finding the required information. So we have desktop search tools which index the data in one's personal computer. Information explosion and need to find relevant information from a huge collection is driving the improvements in Information Retrieval field.

One technique to improve the Information Retrieval performance is to provide the users with ways of finding morphological variants of search terms. If, for example, a user enters the term 'stemming' as a part of the query, it is possible that he/she is also interested in variants such as 'stem' and 'stemmer'. This increases the recall of the IR system. Stemming is also helpful in reducing the size of the index as we need not index all the morphological variants of a word. Most of the times it is good enough to index the stem of the word.

Most of the research on informational retrieval has been based on English as the reference language. Though most of the research in IR is language agnostic there are some areas which are specific to language. Stemming algorithms is one such area. Stemmers for different languages have been developed in the recent years. There is however no stemming algorithm publicly available for Tamil language. With the usage of internet reaching far corners of the world people are now able to create and share content in Tamil. Having good NLP tools will help in developing better Information Retrieval systems for

Tamil content and also in many other text and document processing tools. Stemmer is one such basic NLP tool and is part of other complex tools.

The paper is organised as follows: Sections 2 and 3 give a brief overview on the structure of Tamil words and Stemming algorithms respectively. Section 4 discusses the design of the stemming algorithm and section 5 give details about implementation. The evaluation of the algorithm is presented in section 6 followed by related work in section 7 and conclusion in section 8.

2 STRUCTURE OF A TAMIL WORD

From Wikipedia[4]: “Tamil employs agglutinative grammar, where suffixes are used to mark noun class, number, and case, verb tense and other grammatical categories. Tamil words consist of a lexical root to which one or more affixes are attached.

Most Tamil affixes are suffixes. Tamil suffixes can be derivational suffixes, which either change the part of speech of the word or its meaning, or inflectional suffixes, which mark categories such as person, number, mood, tense, etc. There is no absolute limit on the length and extent of agglutination, which can lead to long words with a large number of suffixes, which would require several words or a sentence in English. To give an example, the word pōkamuṭiyātavarkaḷukkāka (போகமுடியாதவர்களுக்காக) means '*for the sake of those who cannot go*', and consists of the following morphemes:

pōka	*muṭi*	*y*	*āta*	*var*	*kaḷ*	*ukku*	*āka*
go	accomplish	word-joining letter	negation (impersonal)	nominalizer *he/she who does*	plural marker	to	for

The aim of the stemming algorithm is to strip these extra constituents and map them to a stem corresponding to the root word. Ideally all the words with same root words should be stripped to same stem.

3 STEMMING ALGORITHMS

A stemming algorithm is a computational procedure which reduces all words with same root (or if prefixes are untouched same stem) to a common form, usually by stripping each word of its derivational and inflectional suffixes[3]. Simply stated stemming algorithms are used to group words that arise from same stem or root. The result of a stemming algorithm need not be identical to the morphological root of the word;it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. The stemming process is also called as *conflation* sometimes. There are several types of stemming algorithms based on their approach, accuracy etc like: Brute force algorithms, Affix stripping algorithms n-gram based algorithms, Lemmatisation algorithms, Stochastic algorithms etc.

3.1 AFFIX STRIPPING ALGORITHM

Affix removal algorithms remove suffixes and/or prefixes from terms leaving a stem. These algorithms sometimes also transform the stem. A simple example of an affix stripping algorithm is the one that remove the plural forms by Harman [5]

```
If a word ends in "ies" but not "eies" or "aies"
Then "ies" -> "y"
If a word ends in "es" but not "aes", "ees", or "oes"
Then "es" -> "e"
If a word ends in "s", but not "us" or "ss"
Then "s" -> NULL
```

Affix removal algorithms can be simple removal or iterative. In an iterative algorithms affixes are removed until no more affixes can be removed.

4 STEMMING ALGORITHM DESIGN

4.1 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 need not lookup any dictionary or do 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 lack of quality corpus to train statistical algorithms.

4.2 Overview of the 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 checks for the current size of the string.

As shown in Figure - 1, 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.

4.3 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.

Figure 1 - Flowchart of stemming algorithm for Tamil

4.4 Fixing the ending

When a suffix joins a root word one of the following 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. Eg.

மரத்தின்	மரத்த்	மரம்
original word	suffix removed	end recoded

The fix_ending removes the conjunctions and recodes the transformed letters.

4.5 Suffix removal

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

4.5.1 Question suffixes

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

கண்ணனா Is it Kannan	கண்ணன் Kannan

4.5.2 Conjunction suffix

This routine removed the suffix உம்

அவனும் Him and	அவன் Him

4.5.3 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

4.5.4 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 pre-positions (e.g., 'in', 'to', 'for', 'from', etc.).

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

4.5.5 Plural suffix

The plural suffix in Tamil is கள்.

மரங்கள்	மரம்
Trees	Tree

4.5.6 Imperative suffixes

These are used to command a person.

காண்பி	காண்
show me	see

4.5.7 Tense suffixes

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

பிரிகின்றன	பிரி
leaving	leave

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

4.6 Minimum length criteria

Being a strong stemmer it has a tendency to overstem some words to single letters. To prevent this every routine checks 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 actullay only verifies the number of codepoints 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.

5 ALGORITHM IMPLEMENTATION

The algorithm described in the previous section was implemented using Snowball language. Snowball[2] 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 [6] 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 implemetation errors and it widely used now for developing stemming algorithms. Stemming algorithms for many languages like German, French, Turkish etc have been implemeted using the Snowball language.

The code written in Snowball cannot be used with other programs as such. We use snowball compiler to convert the Snowball code to any other programming language. As of now Snowball supports C and Java output. Using the generated C code we can create bindings for many other programming languages. The algorithm implemetor is now not bothered about implemeting algorithms in various other programming languages. It is also possible to extend the snowball compiler to generate code in other programming languages also.

6 EVALUATION

6.1 Tamil WordNet

Tamil WordNet[8] is an attempt to build a lexical network for Tamil language along the lines of the English WordNet so that it can be used as a tool for enhancing the performance of MT systems involving Tamil. The wordnet data is available for free as a sql dump. It has more than 4lakh words with its morphological root. The data available is coded in english transliteration. A transliterator program was written to convert it to UTF8 data. Some of the issues with data are the typing errors and inclusion of many foreign words and non classical Tamil words.

Figure 3 - Database Schema of Tamil WordNet

6.2 Correctness of the algorithm

The correctness of the algorithm is usually measured by identifying the number of semantically related words that are correctly assigned to the same conflation class. Another measure is to see how close the stem is the morphological root of the word.

6.2.1 Variation with morphological root

The stemming algorithm was run against the collection in Tamil WordNet. The Hamming distance between the output of the stemming algorithm and the morphological root was measured. The results are listed in the table below.

Table 1 - Hamming distance from morphological root

Measure	Value
Mean	1.9237
25th percentile	0
Median	2
75th percentile	3

6.2.2 Stems per morphological root

In this test we measure the number of stems per morphological root. The test counts the number of stems created for the words derived out of same root. This describes how correctly we assign the conflation class for the inflated words. Higher number indicates that the algorithm is not stemming the semantically related words to same conflation class.

Table 2 - Stems per morphological root

Measure	Value
Mean	1.7383
25th percentile	1
Median	1
75th percentile	1

For 75% of the root words the number of stems is 1. So the stemmer is doing a good job of mapping similar words to same conflation class.

6.3 Strength of the algorithm

The amount of change the algorithm causes to the given string decides the strength of the algorithm. A strong algorithm tries to remove as many suffixes as possible. A light stemmer usually handles less cases and does not make much modifications to the provided string.

In their paper "Strength and Similarity of Affix Removal Stemming Algorithms", Frakes and Fox[7] propose the following metrics to measure the strength of affix removing algorithms:

1. Mean number of words per conflation class - average number of words that correspond to the same stem for a corpus.
2. Index compression factor - this is the fractional reduction in the index size achieved by stemming. This is given as
3. The number of words and stems that differ - stemmers may often leave words unchanges. This measures such words
4. Mean number of characters removed in forming stems
5. Median and mean modified Hamming distance between the words and their stems - Hamming distance between strings of equal length is the number of character they are differing at the same position. For the strings of unequal length the Hamming distance is the difference in their lengths are also added up.

Table 3 - Modified Hamming Distance Descriptive Statistics

Mean	2.76
Std. deviation	1.97
Minimum	0
25th percentile	2
Median	3
75th percentile	4
Maximum	18

Table 4 - Modified Hamming Distance Descriptive Statistics of some popular stemming algorithms for English Frakes and Fox [7])

	Lovins	Paice	Porter	S-removal
Mean	1.72	1.98	1.16	0.03
Std. deviation	1.64	1.92	1.40	0.19
Minimum	0	0	0	0
25th percentile	0	0	0	0
Median	1	2	1	0
75th percentile	3	3	2	0
Maximum	10	13	9	3

Table 5 - Strength description statistics for Tamil stemmer

Mean Modified Hamming Distance	Median Modified Hamming Distance	Mean Characters Removed	Compression Factor	Mean Conflation Class Size	Word and Stem Different
2.76	3	2.4	0.65	2.88	86.53%

Table 6 - Strength description statistics of some popular stemming algorithms for English (Frakes and Fox [7])

	Mean Modified Hamming Distance	Median Modified Hamming Distance	Mean Characters Removed	Compression Factor	Mean Conflation Class Size	Word and Stem Different
Lovins	1.72	1	1.67	0.29	1.42	69.4%
Paice	1.98	2	1.94	0.33	1.49	69.5%
Porter	1.16	1	1.08	0.17	1.20	56.2%
S-Removal	0.03	0	0.03	0.01	1.01	3.3%

From the metrics measured above it is evident that the designed stemmer is a strong stemmer and is decently accurate. The data for English stemmers is provided as an information and not for comparison. An apple-to-apple comparison cannot be made since the algorithm is for English and uses a different corpus.

6.4 Shortcomings

Because of the agglutinative nature of the language it is possible to form compound words which combine two or more stems into a single word. In such scenarios we will have to find the word boundaries and identify the individual stems. The algorithm proposed in this paper does not handle such scenarious and is only applicable to non-compound words.

7 RELATED WORK

There is a paper published by Vivek Anandan Ramachandran and Ilango Krishnamurthi [9] on an iterative suffix stripping stemming algorithm for Tamil. The paper is behind a paywall and no further details are available regarding the implementation and there is no openly available implementation of the algorithm.

8 CONCLUSION

This paper described the implementation of a rule based iterative affix stripping algorithm and its implementation using Snowball language. The evaluation of the algorithm was done using the Tamil WordNet corpus. The correctness of the algorithm was measured in following ways: 1. Hamming distance between the stem and morphological root. 2. Number of stems per root. The values obtained for these measures indicate that the designed algorithm is reasonably accurate. The strength of the algorithm was measure using the methods proposed by Frakes and Fox[7]. The values obtained in these tests indicate that the designed algorithm is a strong stemmer and it provides good index compression ratio. The The implementation of the algorithm is available in open source and can be used by other tools that require Tamil language stemming.

References

- https://github.com/rdamodharan/tamil-stemmer
- http://snowball.tartarus.org/
- Lovins, Julie Beth. "Development of a Stemming Algorithm." Mechanical translation and computational linguistics 11 (1968): 22-31.
- http://en.wikipedia.org/Tamil_grammar
- Harman, D. "How effective is suffixing?"Journal of the American Society for Information Science.42 (1991): 7–15
- Porter, M. F. "An Algorithm for Suffix Stripping." Program 14 (1980): 130-137.
- Frakes, William B. and Christopher J. Fox. "Strength and similarity of affix removal stemming algorithms". ACM SIGIR Forum 37 (2003): 26-30.
- http://www.au-kbc.org/research_areas/nlp/projects/tamil_wordnet.html
- 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