5th SASTech
Iran | Mashhad
May 12 - 17 | 2011
5th Symposium on Advances in Science & Technology

A New Method for Stemming in Persian Language Considering Exceptions

Somayye Estahbanati, Department of Computer Engineering
Islamic Azad University Science and Research Branch
Ahvaz, Iran
s.estahbanati@gmail.com
Reza Javidan, Department of Computer Engineering
Islamic Azad University – Beyza Branch
Fars, Iran
reza.javidan@gmail.com
Mashalla Abbasi Dezfooli, Department of Computer Engineering
Islamic Azad University Science and Research Branch
Ahvaz, Iran
abbasi_masha@yahoo.com

Paper Reference Number: 15
Name of the Presenter: Somayye Estahbanati

Abstract

In this paper a new algorithm for stemming in Farsi language is presented. This stemmer is based on removing the suffixes and prefixes but a database is used to save the exceptions to decrease error rate. In the proposed method the speed of stemmer and also the percentage of errors are improved. The evaluation results on a small Farsi document collection show significant improvement in precision/recall.

Key words: Stemming, algorithm, Farsi language, Persian Language

1. Introduction

Stemming is a fundamental step in processing textual data preceding the tasks of information retrieval, text mining, and natural language processing. The common goal of stemming is to standardize words by reducing a word to its base. In languages with very little inflection such as English and Mandarin Chinese, the stem is usually not distinct from the "normal" form of the word. However, in other languages, stems are more noticeable. For example, the English verb stem *eat* is indistinguishable from its present tense (except in the third person singular) (Kashif Riaz 2007). There is much research of the effects of stemming on searches of English document collections (Kazem Taghva et al. 2005). Stemmers such as

the Lovins and Porter stemmers sometimes improve precision/recall scores (David A. Hull 1995). However, they only stem English terms.

Farsi or *Persian* is an Indo-European language, spoken and written primarily in Iran, Afghanistan, and a part of Tajikistan. It is associated with Persian culture and is often called *Persian* (Kazem Taghva et al. 2005). Like English, Farsi has affinitive morphology. In other words, suffixes and prefixes are concatenated to words to modify meaning. Farsi is read from right to left, so that prefixes are attached to the right of the root, and suffixes are attached to the left. Like English nouns, Farsi nouns are modified to signify possession, agency and plurality. However, Farsi verbs are modified more extensively than English verbs. Farsi verb forms vary according to tense, person, negation, and mood. The dozens of variations of each Farsi verb are a primary motivation for that stemmer (Kazem Taghva et al. 2005). To facilitate the information retrieval in Farsi search and display technology project (Kazem Taghva et al. 2003), Kazem Taghva, Russell Beckley, and Mohammad Sadeh designed and implemented a Farsi language stemmer (Kazem Taghva et al. 2005). Its aim was to stem a word to find a more general form of it, possibly its root. For example, stemming the term *interesting* may produce the term *interest* or "interes". Though a stemmer might not always give the root, that algorithm want all words that have the same stem to have the same root. On the other hand, for information retrieval, that stemmer do not always wants all words with a given root to have the same stem because some words with the same root may be topically uncorrelated e.g. *preside* and *president*.

In this paper a Farsi algorithm which is based on morphology is described (like porter algorithm in English). The algorithm is implemented and its problems were found. So these problems were solved by presenting an improved algorithm. Finally the results of first algorithm and improved algorithm were compared. The results of improved algorithm were better.

The paper is organized as follows: Section two describes related works have been done in this field .In section three Persian morphology is described and in Section four Farsi stemming algorithm is proposed. Section five describes our proposed implementation method, while in Section six the results of algorithm and improved algorithm are compared. Finally in Section seven conclusion and future works are outlined.

2. Related Works

Most stemming approaches are based on the target languages morphological rules (e.g., the Porter stemmer for the English language (Porter, M 2001)) where suffix removal is also controlled by quantitative restrictions (e.g.,' ing' is removed when the resulting stem has more than three letters as in "jumping," but not in "king") or qualitative restrictions (e.g., '-ize' is removed if the resulting stem does not end with '-e' as in "seize"). Certain ad hoc spelling correction rules can also be applied to improve conflation accuracy (e.g., "running" gives "run" and not "runn"), particularly when phonetic rules are applied to facilitate easier pronunciation. Another approach consults an online dictionary to obtain better conflation results (J. Savoy 1993), while Xu & Croft suggest a corpus-based approach that more closely reflects the language use rather than all its grammatical rules (J. Xu and B. Croft 1998). Few stemming procedures1 have been suggested for languages other than English. The proposed stemmers usually pertain to the most popular languages and some of them, like the Finnish language (S. Tomlinson 2004 , Porter F.M. 1980), seem to require a deeper morphological analysis to achieve good retrieval performance (T. Korenius et al. 2004). Algorithmic

stemmer ignores word meanings and tends to make errors, usually due to over-stemming (e.g., "organization" is reduced to "organ") or to under-stemming (e.g., "create" and "creation" do not conflate to the same root).

Most of the studies so far have been involved in evaluating IR performance for the English language, while studies on the stemmer performance for less popular languages are less frequent. For example, Tomlinson (S. Tomlinson 2004) evaluated the differences between Porter's stemmer (Porter, M 2001) strategy and lexical stemmers (based on a dictionary of the corresponding language) for various European languages. For the Finnish and the German language, lexical stemmer tends to produce statistically better results, while for seven other languages performance differences were insignificant (Ljiljana Dolamic and Jacques Savoy 2009).

There are two famous stemming algorithms in Farsi language:

Kazem Taghva algorithm

This one is like the Porter algorithm in English (Porter, M 2001), which is based on removing the suffix and prefix. Kazem taghva, Russel Beckley and Mohammad Sadeh designed this stemmer in 2005 (Kazem Taghva et al. 2005). In this algorithm Farsi language morphology and a BNF machine whit 40 step are used to remove suffix and prefix.

Krovetz improved algorithm in Farsi

The second algorithm is designed by GholamReza Ghasem Sani and Reza Hesamifard (GholamReza Ghasem Sani and Reza Hesami 2006). The second method is based on the database's information. In the other word all the stems of the language should be saved. At first the input word should be searched in the database, if it is found, the word will be returned as a stem, otherwise the suffixes and prefixes should be removed and it should be searched again in database. This method has some problems. The database needs to be update and also the speed of the stemmer is low.

3. Persian Morphological Descriptions

Persian is a language, spoken and written in Iran, Afghanistan, and a part of Tajikistan. It is written from right to left in the Arabic-like alphabet. In Persian, verbs involve tense, number and person. For example1, the verb "می بینم" (*mi-binam:* I see) is a present tense verb consisting of three morphemes. "م" (*am*) is a suffix denoting first single person "بین" (*bin*) is the present tense root of the verb and "می" (*mi*) is a prefix that expresses continuity.

Rule	Example
می+ بن مضارع + شناسه مضارع (present person identifier + present root + mi)	می بینم *(mi-bin-am)* (I see)
بن ماضی+ ه + بود + شناسه ماضی (past person identifier + bud +eh + past root)	گفته بودم *(goft-e bud-am)* (I had told)
ب + بن مضارع (present root + b)	بمان *(be-män)* (stay)
بن ماضی + ه + شد (shod + h + past root)	گفته شد *(goft-e šod)* (it was told)

Table1. Some rules for verbs in Persian

If a verb has any object pronoun, it can be attached to the end of the verb such as "می‌بینمش" (*mi-bin-am-aš*: I see it) in which "ش" (*aš*: it) is an object pronoun. Also, negative form of verbs is produced with adding "ن" (*ne*) to the first of them. For example, "نمي‌بينم" (*ne-mi-bin-am* - I don't see) is the negative form of the verb "می‌بینم" (*mibinam* - I see). There are some rules for making verbs in Farsi language that some of them are shown in table (Table 1).

Joining	**Result noun**
کتاب + ها *(hä + ketäb)* (hä + book)	کتاب ها *(ketäb-hä)* (books)
درخت + ان *(än + deraxt)* (*än* + tree)	درختان *(deraxt-än)* (trees)
(Mokassar form) نسخ *(nosakh)* (prescription)	نسخ *(nosakh)* (prescription)
دانا + ی+ ان *(än + y + dänä)* (än + y +wise)	دانایان *(dänä-yän)* (wise people)

Table2. Some kinds of plural form in Persian

There are many challengeable rules for nouns that in following, one of them is described. The plural forms of nouns are formed by adding the suffixes (**ها, ان, ات, ون, ین**). "**ها**" (*hä*) is used for all words. "**ان**" (*än*) is used for humans, animals and everything that is alive. Also, "**ات , ون, ین** " (*ät ,un , in*) is used for some words borrowed from Arabic and some Persian words. There are another kind of plural form in Persian that is called *Mokassar* which is a derivational plural form (irregulars in Persian). Some examples of plural form are shown in table (Table 2).

Also, there are some orthographic rules which show the effects of joining affixes to the word. For example, consider there are two parts of a word: A and B for joining as BA (Consider, Persian is written right to left). If the last letter of A and the first letter of B are "ا" (*ä*), one letter "**ی**"(*y*) is added between them. Assume A is "**آقا**" (*äghä* - mister) and B is "**ان**" (*än*), the joining result is "**آقایان**" (*äghä-yän:* men) (Amir Azim Sharifloo and Mehrnoush Shamsfard).

4. The Algorithm

Our Farsi stemmer is based on morphology and uses multiple phases conforming to the rules of suffix stacking. Also, it enforces a lower bound on the information a stem retains. The Farsi stemmer uses stem length to define a lower bound on information content (the minimum stem length is three). This limit is crucial when a non-suffix substring of a short word is incorrectly identified as a suffix. The Farsi stemmer identifies prefixes, and it removes prefix according to defined sequences.

The first step of the stemmer algorithm is to find a terminal substring of the input word that is in a list of common Farsi morphological prefix. Then it removes the suffix of

input word. If multiple suffixes match the word, the stemmer chooses the longest suffix that would leave a stem with three or more characters. Consider the Farsi word دستشان (”their hands”). Both the plural suffix ان and the plural possessive شان match the end of the word. Removing ان leaves four letters, and removing شان leaves three letters. Because both leave long enough stems, the stemmer removes شان the longest, giving دست (hand).

The suffixes are grouped as *verb-suffixes*, *plural-noun-suffixes*, *possessive-noun-suffixes*, *other-noun-suffixes* (e.g. نده), and *other-suffixes* (e.g. تر). This grouping guides removal of prefixes from verbs and removal of multiple suffixes from a noun. If the stemmer first identifies the suffix ند in the word نرفتند (”they did not go”) as a verb-suffix, it then identifies and removes the prefix ن to produce the stem رفت (”went”). For example, the stemmer first finds the possessive noun suffix یمان in the word خواننده هایمان (”our singers”), then it finds the plural noun suffix ها and, finally, it finds the other-noun-suffix نده (which signifies agency) to give the stem خوان (”sing”). Hence the stemmer removes up to three suffixes from nouns.

In addition, there are some unusual cases. Usually, when the stemmer finds the suffix تان, it removes it. However, when it is preceded by س. it ignores the suffix, because the Farsi suffix ستان (”location of”; pronounced ”stan”) is often used for countries and regions, e.g. ”Kurdistan.”. The stemmer does not remove ستان because generally, the resulting conflations (e.g. Kurd = Kurdistan) are not helpful for a search engine.

Another exception is that the stemmer finds verbal suffixes د and ت but does not remove them. That the infinitives end with دن or تن. Most of the Farsi tenses are formed after removing the suffix I but leaving characters د or ت. In many cases, the stemmer looks at the letter preceding a supposed suffix. Often, this pre-suffix can be used to determine whether the match is actually a suffix and, if it is, whether it ought to be removed. In such cases, if the suffix is removed, the pre-suffix remains (C. Peters et al. 2008). Our first algorithms results had some problems because of the exceptions. These exceptions should be found out to improve the algorithm.

There are some words that are structurally similar to other words. These words should not be used by prefix stemmer and suffix stemmer. For example the first letter of non-verb word “برنامه” is “ب” which is same as the prefix of imperative verbs in Farsi. But the letter “ب” should not be removed. Or the word “نیمکت” starts with “ن” which is similar to negative verbs and it ends with “ت” that is same as possessive pronoun. But these letters should not be removed as prefix and suffix.

Also there are some plural words in Farsi, named Mokassar which there are no certain rules to make them. Current rules couldn’t be used to find these words stem.

Furthermore this algorithm has a restriction which the resulted stem should have three or more letters. But there are some words that their stem's length is less than three. For example for the verb “**می‌کنیم**” the algorithm removes the term “**می**” as prefix. Then it detects “**یم**” as the longest suffix, but if the algorithm removes “**یم**” the remind part will have only two letters. So it removes just “**م**” and returns “**کنی**” as the stem, while the correct stem is “**کن**”.

So a data base is used to save these words stem and the algorithm is improved by considering these exceptions.

5. Implementation

The BNF machine is used to implement the algorithm. This implementation includes a suffix stemmer and a prefix stemmer. All suffixes will be removed during the fifteen states of

the suffix stemmer. Also the prefix stemmer has two states to detect and remove the prefixes. This implementation has two final steps that will be described later. To save the detected suffixes and prefixes of each word to compare the class of suffixes or prefixes whenever it needs, two arrays are used. Suffix stemmer receives the word in reverse direction. After some proportional steps one of these following final states will be observed:

1) State0: in this state a suffix or prefix has been detected. So it will be removed and the word will be given back to the suffixstemmer or prefixstemmer as a new word.
2) Last state: the above operation is repeated until it can't detect any suffix or prefix or the word contains less than three letters. In this case the word is returned without any removal.

Prefix stemmer acts like suffix stemmer but it doesn't need to reverse the word. Before removing any suffix and prefix in each stage, the stemmer checks the suffixes and prefixes that were removed in previous steps and also it checks the type of the word. The current suffix or prefix will be removed, if its type is similar to previous removed suffixes and prefixes and it should be consistent whit the type of the word. Also a data base is used to improve the stemmer. In this database, non-verb words which start with a term that is similar to a verb maker prefix or words which end with a term that is similar to a suffix are saved. But if after removing these terms, the remained part of word had less than three letters, these words should not be saved in database.

For example the word "**میوه**" starts with "**مي**" which is similar to verb maker prefix "**مي**" in Farsi. But it's not a prefix. If the part "**مي**" is removed, the remained part "**وه**" will have only two letters. So this word should not be saved in database. Or the word "**نیما**" starts with "**ن**" which is similar to negative verb's structure. If the term "**ن**" is removed, the remained part will have three letters. Therefore the word "**نیما**" should be saved in database. Also some plural words named mokassar and their singulars are saved in database. At the start of algorithm, the word should be searched in database. If it is found, its stem will be returned. Otherwise it will be used by algorithm's functions to remove suffixes and prefixes.

Furthermore, some words which their stems have less than three letters are saved in database. If after removing the suffixes or prefixes the stemmer confronts a stem with less than three letters, at first it will search the database. If the stem is found in the data base, it will be returned. But if it isn't found, the stemmer doesn't remove the suffix or prefix.

6. Evaluation

Four texts are selected with various topics on internet to test the algorithms. The results are shown in table and figure (Table 3 and Figure 1).

Test number	Percentage of Correct results (first algorithm)	Percentage of Correct results (improved algorithm)	Time (first algorithm)	Time (improved algorithm)
1	82.00	98.00	2 sec	2 sec
2	79.85	97.01	4 sec	4 sec
3	81.07	97.14	16 sec	16 sec
4	85.60	97.21	31 sec	32 sec

Table 3. Results of comparison

As mentioned before, to improve our algorithm a data base is used that some exceptions are saved in it. This evaluations show that the percentage of correct results is increased while the speed of algorithm doesn't change.

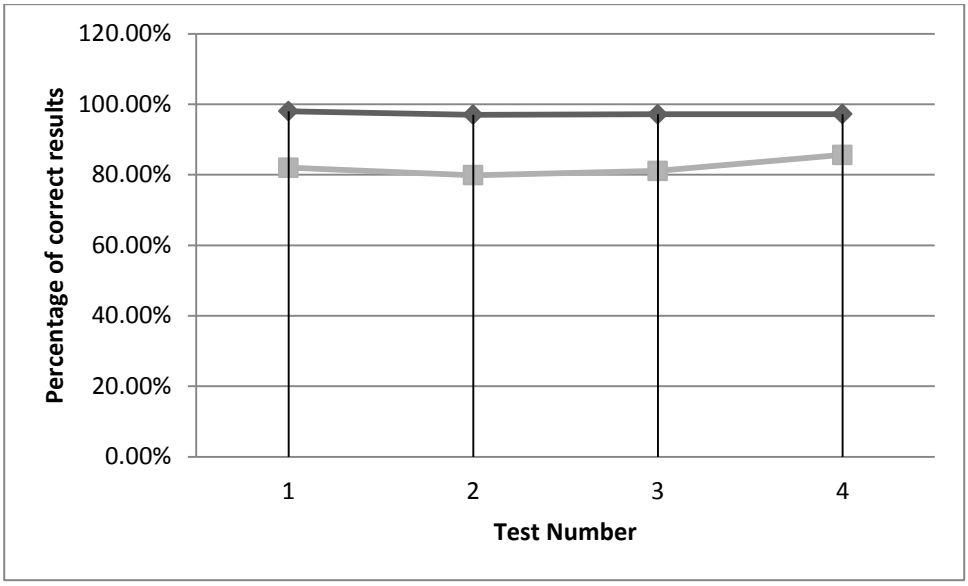

Fig 1: Chart of result

7. Conclusion And Remarks

In this paper stemming problem briefly is described. Then the applications of stemming and different types of stemming algorithms were explained. At first an algorithm is implemented based on morphology and its problems were described. Afterward a modified algorithm was presented to improve the results. In the proposed method a database is used which contains some exceptions and based on morphology. Morphology is used to find the stem of the words. The stemmer was improved by saving the words that are similar to other words structure, and also some exceptional plural words named Mokassar and some stems that have less than three letters in a database. The number of these words is low in compare with the number of all Farsi words. But this algorithm is depended on database and in some cases the result is wrong because the stemmer can't detect the type of the words. This problem will be solved by finding out the type of the words according to the structure of the sentences.

References

Ljiljana Dolamic and Jacques Savoy (2009) Persian Language, is Stemming Efficient?

GholamReza Ghasem Sani and Reza Hesami (2006) A stemming algorithm for Farsi language

David A. Hull (1995) Stemming algorithms case study for detailed evaluation

T. Korenius et al. (2004) Stemming and lemmatization in the clustering of finnish text documents

C. Peters et al. (2008) Advances in Multilingual and Multimodal Information Retrieval

Porter, M (2001) A language for stemming algorithms

Porter F.M. (1980) An algorithm for sux stripping

Kashif Riaz (2007) Challenges in Urdu Stemming (A Progress Report)

Eiman Tamah Al-Shammari (2008) TOWARDS AN ERROR-FREE STEMMING

J. Savoy (1993) Stemming of French words based on grammatical category

Amir Azim Sharifloo and Mehrnoush Shamsfard A Bottom up Approach to Persian Stemming

Kazem Taghva et al. (2005) A Stemming Algorithm for the Farsi Language

Kazem Taghva et al. (2003) Farsi Searching and Display Technologies

S. Tomlinson (2004) Lexical and algorithmic stemming compared for 9 European languages with Hummingbird SearchServerTM at CLEF 2003

J. Xu and B. Croft (1998) Corpus-based stemming using cooccurrence of w,ord variants