

Implementation of a New Method for Stemming In Persian Language

Asieh Estahbanati

Dep. of Computer Engineering
Islamic Azad University
Khoosestan Science and
Research Branch
Tel: +989173384722

s.estahbanati@gmail.com

Reza Javidan

Dep. of Computer Engineering
Islamic Azad University
Beyza Branch
Fars - Iran
Tel: +989173159656

reza.javidan@gmail.com

Mashalla Abbasi Dezfooli

Dep. of Computer Engineering
Islamic Azad University
Khoosestan Science and
Research Branch
Tel: +989163009266

abbasi masha@yahoo.com

ABSTRACT

In this paper, a new stemmer algorithm for the Persian language is implemented. It is based on Kazem Taghva algorithm. The evaluation results of the proposed method on the small Farsi document collections are acceptable. But some problems in the morphological based stemmers in Farsi language are found. So some methods are proposed to solve these problems.

Categories and Subject Descriptors

D.2.2 [Stemming Algorithm]: Language – *root*, *word*, *structures*.

General Terms

Algorithms, Design, Experimentation, Languages, Theory.

Keywords

Stemming, algorithm, Farsi language, Persian Language.

1. INTRODUCTION

A *morpheme* is the smallest language unit that carries a semantic interpretation. Morphemes are generally a distinctive collocation of phonemes (the free form *pin* or the bound form -s of *pins*) having no smaller meaningful members. A *stem* in linguistics is the combination of the basic form of a word (called the root) plus any derivational morphemes, but excluding inflectional elements. Some researches make a distinction between the concepts of *stem* and *root* but the difference has no practical value for stemming purposes. Alternatively, a stem is the form of the word to which inflectional morphemes can be added [1].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WIMS'11, May 25-27, 2011 Sogndal, Norway
Copyright © 2011 ACM 978-1-4503-0148-0/11/05

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.

Inflection refers to modification of a word. More precisely, a word is modified so that it reflects grammatical information such as word gender, tense, or person. The root of the English verb form *destabilized* is *stabil-* (alternate form of *stable*); the stem is *de•stabil•ize*, which includes the derivational affixes *de-* and *-ize*, but not the inflectional past tense suffix *-(e)d*.

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)[3]. There is much research of the effects of stemming on searches of English document collections [4]. Stemmers such as the Lovins and Porter stemmers sometimes improve precision/recall scores [6]. 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* [4].

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 [4].

To facilitate the information retrieval in Farsi search and display technology project [5], Kazem Taghva, Russell Beckley, and Mohammad Sadeh designed and implemented a Farsi language stemmer [4]. 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*.

There are same stemming algorithms for Farsi language. However, in this paper, Kazem Taghva's algorithm is used as the basis and it was changed to improve the performance of the algorithm. Here our implementation is simpler than the previous implementation of taghva algorithm. Our experiments on prototype data showed the effectiveness of the proposed method.

The paper is organized as follows: Section two describes related works have been done in this area. In Section three Kazem Taghva farsi stemming algorithm is described. Section four describes our proposed implementation method, while in Section five evaluates the stemmer according to precision/recall scores. Finally Section six discusses our conclusion and future works.

2. RELATED WORKS

Most stemming approaches are based on the target languages morphological rules (e.g., the Porter stemmer for the English language [7]) 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 [8], while Xu & Croft [9] suggest a corpus-based approach that more closely reflects the language use rather than all its grammatical rules. Few stemming procedures have been suggested for languages other than English. The proposed stemmers usually pertain to the most popular languages [10], [11] and some of them, like the Finnish language, seem to require a deeper morphological analysis [12] to achieve good retrieval performance.

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 [10] evaluated the differences between Porter's stemmer strategy [7] 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 [13].

So there are two main methods for stemming. The first one is based on the morphological rules of the words and the second

one is based on saving the stems in a database. The first method could be implemented for the languages that have the certain morphological rules. This method doesn't depend on the database and doesn't need to be updated because the morphological rules don't change. In the second method the stems of the words are saved in a database. But this method has some problems. The database needs to be updated and the speed of the stemmer is low because of the large number of searches in the database. This method is useful for the languages that don't have the certain morphological rules.

Farsi language has certain rules for morphology. So the first method could be chosen. There are two famous stemming algorithms in Farsi language:

- 1- Kazem Taghva algorithm
- 2- Krovetz improved algorithm in Farsi

The first one is like the Porter algorithm in English [7], that is based on removing the suffix and prefix, which is thoroughly described in Section three. Kazem taghva, Russel Beckley and Mohammad Sadeh designed this stemmer in 2005 [4]. Their paper explains the algorithm and the results without any implementation. So we need the implementation to evaluate and test those results. If this stemmer has any defect or problem, we try to solve this or design a new algorithm without those problems.

The second algorithm is designed by GholamReza Ghasem Sani and Reza Hesamifard [14]. This method is based on the database's information. In the other word we have to save all the stem of the language. At first we should search in the database, if the input word is found, the word will be return as a stem, otherwise the suffixes and prefixes should be removed and it should be searched again in database.

Due to mentioned causes the Kazem Taghva's algorithm is chosen to impalement and finding its problems and finally proposing new methods to solve these problems.

3. PERSIAN FROM A MOROHOLOGICAL PERSPECTIVE

Persian is an Indo-European language, spoken and written primarily 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 "خوانم می" (*mi-xānam*: I read) is a present tense verb consisting of three morphemes. "م" (*am*) is a suffix denoting first single person "خوان" (*xān*) is the present tense root of the verb and "می" (*mi*) is a prefix that expresses continuity.

If a verb has any object pronoun, it can be attached to the end of the verb such as "میخوانمش" (*mi-xān-am-aš*: I read 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, "نمیخوانم" (*ne-mi-xān-am* - I don't read) is the negative form of the verb "میخوانم" (*mi-xānam* - I read). We have gathered 43 rules for verbs, some of them are shown in Table 1.

Nouns are more challengeable than others in Persian. We have gathered many rules for nouns that in following, we describe one of them. The plural forms of nouns are formed by adding the suffixes (هـا، ان، ات، ون، ين) (*hā*) is used for all words. “ان” (*ān*) is used for humans, animals and everything that is alive. Also, “ات، ون، ين” (*āt, un, in*) is used for some words borrowed from Arabic and some Persian words. We have 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 2 [15].

Also, there are some orthographic rules which show the effects of joining affixes to the word. For example, consider that we have 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 “ی” (*ā*), one letter “ی” (*y*) is added between them. Assume A is “دانا” (*dānā* - wise) and B is “ان” (*ān*), the joining result is “دانیان” (*dānā-yān*: wise people) [15].

Table 1. Some rules for verbs in Persian

Rule	Example
می + بن مضارع + شناسه مضارع (present person identifier + present root + mi)	می خوانم (<i>mi-xān-am</i>) (I read)
بن ماضی + ه + بود + شناسه ماضی (past person identifier + bud +eh + past root)	رفته بودم (<i>raft-e bud-am</i>) (I had gone)
ب + بن مضارع (present root + b)	بگذر (<i>be-gozar</i>) (Pass)
بن ماضی + ه + شد (shod + h + past root)	خوانده شد (<i>xand-e šod</i>) (it was read)

Table 2. Some kinds of plural form in Persian

Joining	Result noun
کشور + ها (<i>hā + kešvar</i>) (hā + country)	کشورها (<i>kešvar-hā</i>) (countries)
درخت + ان (<i>hā + deraxt</i>) (hā + tree)	درختان (<i>deraxt-ān</i>) (trees)
کتاب (Mokassar form) (<i>kotob</i>) (books)	کتاب (<i>kotob</i>) (books)
آقا + ی + ان (<i>ān + y + āghā</i>) (ān + y + mister)	آقایان (<i>āghā-yān</i>) (men)

4. THE ALGORITHM

The Farsi stemmer [4] is similar to the Porter stemmer [7]. Both are based on morphology. Also, both stemmers search for certain suffixes and use multiple phases conforming to the rules of suffix stacking. Furthermore, they both enforce a lower bound on the information a stem retains. However, there are important differences. For example, the Porter stemmer identifies patterns of consonants and vowels to estimate the information content; in Farsi, many spoken vowels are not written, so the Farsi stemmer uses stem length to define a lower bound on information content (currently, 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. Another difference is that the Farsi stemmer identifies prefixes while the Porter Stemmer does not.

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 suffixes. 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”). Noun suffixes are stacked according to the pattern (reading right-to-left):

$\{Possessive\}\{Plural\}\{Other\} < Stem >$

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 we believe, generally, the resulting confluations (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 [1].

5. IMPLEMENTATION

We used the BNF machine to implement this algorithm. This implementation includes a suffixstemmer and a prefixstemmer. The suffixstemmer has 15 states and it detects and removes all suffixes during these states. The prefixstemmer has 2 state for detection and removal of all prefixes.

We have two final states in this implementation that will be described later. Also we use two array for saving the detected suffixes and prefixes of each word to compare the class of suffixes or prefixes whenever it needs.

Suffix stemmer receives the word in reverse direction. After some proportional steps we have one of these following final states:

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. Laststate: 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 with the type of the word.

6. EVALUATION

We use Common Text on internet, to test the algorithm. The results are shown in Table 3 and Figure 1.

Table 3. Results

Test number	Total words	Correct results	Incorrect result	Percent correctly
1	41	33	8	80.50
2	89	76	13	85.40
3	120	97	23	80.84
4	130	108	22	83.07
5	547	492	55	89.94
6	1109	994	115	89.63

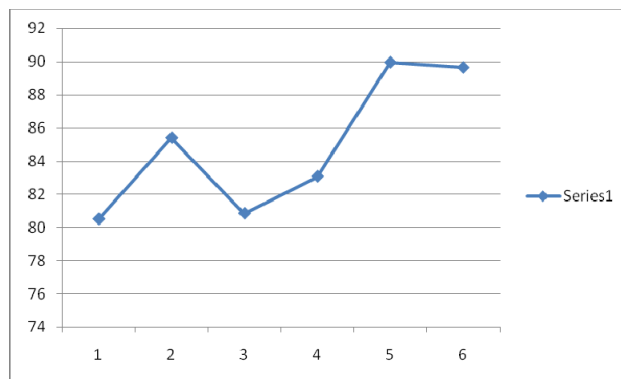


Figure 1. Results and Comparison

We select six texts with various topics on internet, and test them with our algorithm. As it can be seen, by increasing the quantity of tested words, the average percentage of errors is decreased.

You can see the result of a stemming in Table 4. The first column contains the words of the selected text and the second column contains the roots that are found by our stemmer. The third column shows the correct root of the word.

As it can be seen, the results are acceptable. It should be noted that many of these errors are due to non-verb words that are structurally similar to verbs or because of the verbs that their stems are less than three letters.

Table 4. Word and Stem

Correct stem	Stem	Word
1- ویژہ	1- ویژہ	1- ویژگیهای
VB -2	VB -2	VB -2
Net -3	Net -3	Net -3
4- در	4- در	4- در
5- زمینہ	5- زمین	5- زمینہ
6- برنامه	6- رنامه	6- برنامه
7- نویس	7- ویسی	7- نویسی
8- اینترنت	8- اینترن	8- اینترنت
9- و	9- و	9- و
10- پایگاه	10- پایگا	10- پایگاه
11- داده	11- داد	11- داده
12- به	12- به	12- به
13- نحو	13- نحو	13- نحو
14- قابل	14- قابل	14- قابل
15- توجہ	15- توج	15- توجہی

16- بهبود	16- هیود	16- بهبود
17- یاب	17- یافته	17- یافتہ
18- و	18- و	18- و
19- این	19- این	19- این
20- مزیت	20- مزای	20- مزایای
21- زیاد	21- زیاد	21- زیادی
22- را	22- را	22- را
23- برای	23- رای	23- برای
24- برنامه	24- رنامه	24- برنامه
25- نویس	25- ویس	25- نویسان

7. CONCLUSION AND REMARKS

In this paper, at first we gave a brief description of the word 'stemming'. Then the usage of stemming algorithms was described and the available stemming algorithms were mentioned in brief. After that Kazem Taghva's farsi algorithm and improved Korvetz algorithm were checked and Finally the morphological based algorithm was chosen to implement because the Farsi language has the certain morphological rules and the speed of this algorithm is better and also it doesn't depend on database and doesn't need to be updated.

In this paper a new algorithm is implemented based on the Kazem Taghva's algorithm. But the number of states in the new algorithm was less than the Taghva's.

As it mentioned before, we used morphology to find the stem of the words. We had some problems during the stemming that most of them were due to lack of diagnosis of the words types.

We propose two methods to solve these problems:

- 1- By identify the types of the words according to the grammar
- 2- By saving the words that are similar to defined structures and also the verbs that have a stem with less than three letters in a database. The Quantity of these words is too low compared with all Persian words

If we use the second method to improve the algorithm, it can be said that Porter algorithm and Korvetz algorithm are combined together and the problems of these two algorithms will be solved.

We didn't do it due to lack of the time.

8. REFERENCES

[1] Porter, M. 2001. A language for stemming algorithms. DOI= <https://snowball.tartarus.org/texts/introduction.html>.

[2] Tamah Al-Shammari, E. 2008. Towards an Error free Stemming. In *Proceeding of IADIS European Conference Data Mining*.

[3] Riaz, Kashif. 2007. Challenges in Urdu Stemming (A Progress Report). In *Proceeding of BCS IRSG Symposium: Future Directions in Information Access (FDIA 2007)*.

[4] Taghva, Kazem, Beckley, R. and Sadeh, M. 2005. A Stemming Algorithm for the Farsi Language. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume I - Volume 01* Pages: 158 – 162.

[5] Taghva, Kazem, Young, Ron, Coombs, Jeffrey, Beckley, Russell, Sadeh, M., and Pereda, Ray. 2003. Farsi Searching and Display Technologies. In *Proceeding of the Symposium. on Document Image Understanding Technology*, pages 4146, Greenbelt, MD.

[6] Hull, David A. 1995. *Stemming Algorithms Case Study for Detailed Evaluation*. Technical Report. Rank Xerox Research Centre, Meylen, France.

[7] Porter., M. F. 1980. An Algorithm for SUX Stripping. *Programs*, 14(3):130137.

[8] Savoy, J. 1993. Stemming of French Words Based on Grammatical Category. *Journal of the American Society for Information Science*, vol. 44, no. 1, pp. 1–9.

[9] Xu, J. and Croft, B. 1998. Corpus-Based Stemming Using Cooccurrence of Word Variants. *ACM-Transactions on Information Systems*, vol. 16, no. 1, pp. 61–81.

[10] Tomlinson, S. 2004. Lexical and Algorithmic Stemming Compared for 9 European Languages with Hummingbird SearchServerTM at CLEF 2003. In *Comparative Evaluation of Multilingual Information Access Systems*, ser. Lecture Notes in Computer Science. vol. 3237. Berlin: Springer-Verlag. pp. 286– 300.

[11] Peters, C., Jijkoun, V., Mandl, T., Muller, H., Oard, D., Peñas, A., and Santos, D. Eds. 2008. *Advances in Multilingual and Multimodal Information Retrieval*. Lecture Notes in Computer Science. Berlin: Springer-Verlag. vol. 5152.

[12] Korenius, T., Laurikkala, J., Jarvelin, K., and Juhola, M. 2004. Stemming and Lemmatization in The Clustering of Finish Text Documents. In *Proceedings of the ACM-CIKM*. Washington DC: The ACM Press. pp. 625–633.

[13] Dolamic, Ljiljana , and Savoy, Jacques 2009. Persian Language, is Stemming Efficient?. In *Proceeding of 20th International Workshop on Database and Expert Systems Application (DEXA '09)*. Pages: 388 – 392.

[14] Ghasem Sani, GholamReza, and Hesami, Reza 2006. A Stemming Algorithm for Farsi Language. In *Proceeding of 11 International CSI Computer Conference (CSICC'2006)*.

[15] Azim Sharifloo, Amir, and Shamsfard, Mehrnoush. *A Bottom up Approach to Persian Stemming*. Shahid Beheshti University, Tehran, Iran.