

# A Stemming Algorithm for the Farsi Language

Kazem Taghva, Russell Beckley, and Mohammad Sadeh  
Technical Report 2003-02  
Information Science Research Institute  
University of Nevada, Las Vegas

August 2003

## Abstract

In this paper, we report on the design and implementation of a stemmer for the Farsi language. The results of our evaluation on a small Farsi document collection shows a significant improvement in precision/recall.

## 1 Introduction

*Farsi*, also known as *Persian*, is an Indo-European language, spoken and written primarily in Iran, Afghanistan, and a part of Tajikistan. As a part of our Farsi search and display technology project[6], we decided to study, design, and build a stemmer specifically for the Farsi language. Our intention is to use this stemmer as a component of our retrieval engine.

To stem a word is to reduce it to a more general form, possibly its root. For example, stemming the term *interesting* may produce the term *interest*. Though the stem of a word might not be its root, we want all words that have the same stem to have the same root. The effect of stemming on searches of English document collections has been tested extensively. In some contexts, stemmers such as the Lovins and Porter stemmer improve precision/recall scores. [2] However, these stemmers are language specific, and to get similar results on a Farsi collection requires a Farsi stemmer.

Like English, Farsi has an affixitive morphology. In other words, suffixes and prefixes are concatenated to Farsi words to modify the meaning. Since Farsi is read from right to left, what appears to be the end of a word to an English reader is actually the beginning. Prefixes might at first appear to be suffixes. Like English nouns, Farsi nouns are affixed to signify possession and plurality. On the other hand, Farsi verbs are modified more extensively than English verbs. Farsi verbs vary form according to tense, person, negation, and mood. Therefore, a given verb may have scores of variations. As a matter of fact, one of the motivations behind our stemmer was the high number of variations for Farsi verbs.

This paper consists of five sections in addition to this Introduction. Section two is a short overview of Farsi and its grammar. Section three is the description of our stem-

Infinitive	Imperative Mood	English Translation
آوردن	آور	to bring
پرسیدن	پرس	to ask

Table 1: Regular Infinitives and Imperative Moods

Infinitive	Imperative Mood	English Translation
رفتن	رو	to go
کردن	کن	to do

Table 2: Irregular Infinitives and Imperative Moods

ming algorithm, Section Four is our implementaion, Section Five is a precision/recall evaluation of this stemmer, and Section Six discusses our conclusion and future work.

## 2 Farsi Language

In the Farsi language, words are usually built up from the imperative forms of the verbs. Hence, from a linguistic point of view, the first step in extracting the root is to find the imperative mood of the word. For example, we can find the root of شنونده (in English “listener”) by removing the suffix نده. In general, obtaining the imperative mood is not easy since there are irregular infinitives. The regular infinitives in Farsi end with the suffix دن and the imperative mood of the regular infinitives are obtained by removing two or sometimes, the last three characters. Examples of regular infinitives and their imperative moods are listed in Table 1.

This regular infinitive and the pattern of obtaining the imperative mood is known as قیاسی (pronounced ghiasi). The irregular infinitives end with characters تن and usually there are no regular patterns for obtaining their imperative form. Examples of irregular infinitives and their imperative moods are listed in Table 2.

The imperative form of irregular infinitives are based on how the words are heard or used and is known as سماعی (pronounced samaii).

In Farsi, it is common to add the prefix ب (in English “b”) and ن (in English “n”) for positive and negative moods. So for example برو (go) and نرو (do not go) are the positive and negative forms of رو respectively.

Assuming one can obtain the imperative form of a verb, then one can follow the grammar rules of Farsi to generate tense such as present tense. So for example to generate the *indicative present tense* (in Farsi مضارع اخباری) for the verb رفتن (to go), we start with the positive imperative برو and add the appropriate suffixes as found in Table 3.

The present tense rules are generally used to generate other tenses. Another group of tenses such as past tense is generated from the infinitives by removing the character ن and adding the same suffixes as above. Hence the past tense (in Farsi زمان گذشته) of the verb رفتن رفت and its variations are listed in Table 4.

It should be noted that no suffix is added for past tense singular third person; this is due to the fact that if we add the suffix د, then the pronunciation becomes awkward.

Suffix	Present Tense	English Translation
م	بروم	I go
ی	بروی	You go
ا	برود	He goes
یم	برویم	We go
اید	بروید	You go
ند	بروند	They go

Table 3: Present Tense Suffixes

Suffix	Past Tense	English Translation
م	رفتم	I went
ی	رفتی	You went
	رفت	He went
یم	رفتیم	We went
اید	رفتید	You went
ند	رفتند	They went

Table 4: Past Tense Suffixes

Readers interested in various forms of Farsi verbs are referred to [1].

Farsi has specific rules for plural, possessive, and comparative forms of nouns. The plural forms of Farsi nouns are obtained by adding the suffixes ها, ان and, for words borrowed from Arabic ات. Table 5 shows examples of plurals.

Farsi has a well defined and detailed grammar. The above description is meant to give the reader a flavor of Farsi morphology.

### 3 The Algorithm

The Farsi stemmer is similar to the Porter stemmer [3] in that each is based on the morphology of its language. Both stemmers match words with a set of suffixes and use multiple phases conforming to the rules of suffix stacking. Furthermore, they 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 stemmer cannot count them. Therefore, the Farsi stemmer uses stem length to define

Suffix	Plural	English Translation
پسر	پسرها	sons
جوان	جوانان	young people
مشکل	مشکلات	difficulties

Table 5: Plural Suffixes

a lower bound on information content (in the current version, minimum stem length = 3). Also, 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 the *list* of Farsi suffixes. This suffix list was prepared by hand using Farsi grammar. If multiple suffixes match the word, the stemmer chooses the longest suffix that would leave a sufficiently long stem. Consider the Farsi word دستشان (in English “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. Since both leave sufficiently long stems, the stemmer removes شان, the longest stem, producing دست (hand).

The list of suffixes is also grouped into *verb-suffixes*, *plural-noun-suffixes*, *possessive-noun-suffixes*, *other-noun-suffixes* (e.g. نده), and *other-suffixes* (e.g. تر). This grouping helps removal of prefixes in the case of verbs and removal of multiple suffixes from a noun. For example the stemmer first identifies the suffix ند in the word نرفتند (in English “they did not go”) as a verb-suffix; it then identifies and removes the prefix ن to produce the stem رفت (in English “went”).

In the case of nouns, suffixes are stacked according to their pattern:

{possessive-noun-suffix}{plural-noun-suffix}{other-noun-suffix}<stem>

For example, the stemmer first identifies the possessive-noun-suffix يمان for the word خواننده هايمان (in English “our singers”), then it identifies the plural-noun-suffix ها, finally it identifies the other-noun-suffix نده to reach the stem خوان (in English “sing”). Hence in the case of nouns, the stemmer makes up to three attempts to remove suffixes.

Some suffixes require treatment that does not conform to the preceding general description. When the stemmer finds the suffix تان preceded by س it ignores the suffix. The Farsi suffix ستان (in English “location of”) is often used for countries and regions, e.g. “Kurdistan.” The stemmer does not modify these words.

Another exception is that the stemmer finds verbal suffixes د and ت but does not remove them. It was explained in Section 2 that the infinitives end with دن or تن. Most of the Farsi tenses are formed after removing the suffix ن but leaving characters د or ت.

## 4 Implementation

To determine which suffix terminates the input word, the Farsi stemmer uses a Deterministic Finite Automata (DFA). The DFA’s input string is obtained by reversing the stemmer’s input string. Every state is an accepting state. Figure 1 is a schematic representation of a portion of the 70 state machine.

The DFA is encoded as a two-dimensional array. The rows represent states and the columns represent input letters. Table 6 shows a two-dimensional array representing the machine in figure 1 in a similar fashion.

The DFA driver starts from the end of the stemmer’s input word and works toward the third letter from the front. The DFA never sees the first two characters of the word.

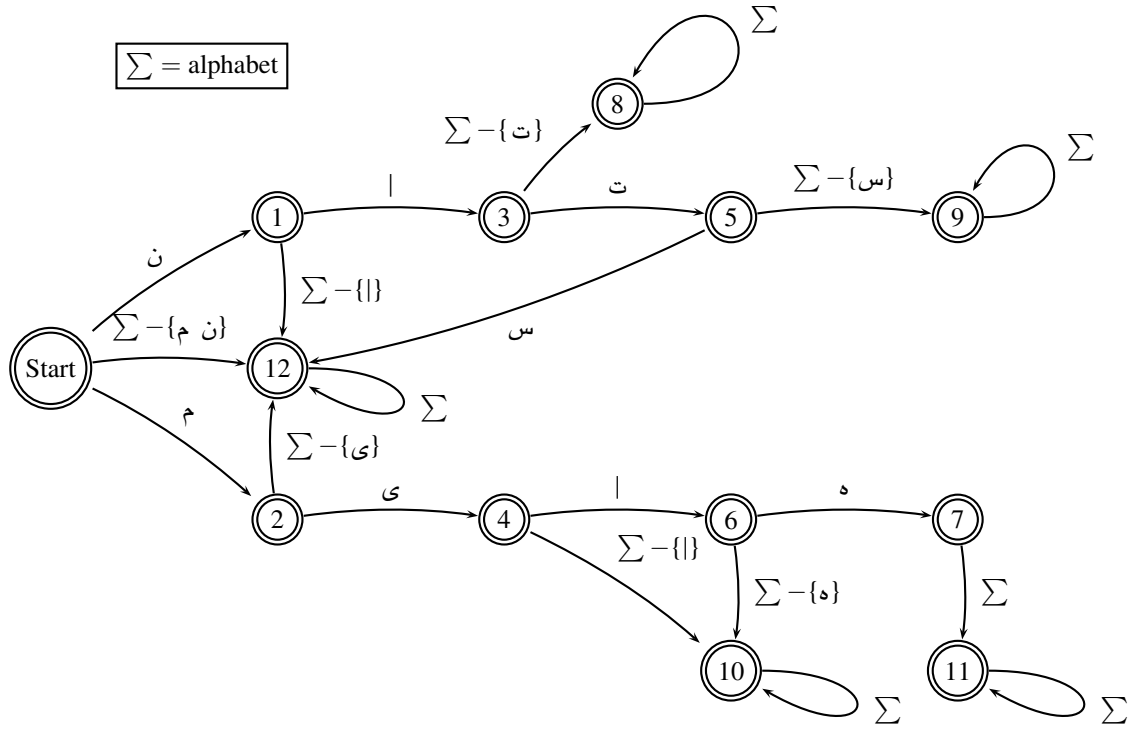


Figure 1: A part of the Farsi Stemming DFA

state	ا	ت	س	م	ن	ه	ی	else
0	12	12	12	2	1	12	12	12
1	3	12	12	12	12	12	12	12
2	12	12	12	12	12	12	4	12
3	8	5	8	8	8	8	8	8
4	6	10	10	10	10	10	10	10
5	9	9	12	9	9	9	9	9
6	10	10	10	10	10	7	10	10
7	11	11	11	11	11	11	11	11

Table 6: Two-dimensional array representing the DFA in Figure 1

state	SuffixGroup[state]
0	NIL
1	NIL
2	NIL
3	NIL
4	NIL
5	PL2
6	VB2
7	VB2
8	PL2
9	PO3
10	VB2
11	VB4
12	NIL

Table 7: SuffixGroup[] for the machine in Figure 1

The idea here is to look for a suffix while guaranteeing the minimum three character stem length requirement. In each round the driver determines the next state by observing the entry at row  $s$  and column  $l$ , where  $s$  is the current state and  $l$  is the input character. When the machine reaches a final state, the word and the actual state is returned to a post processor for suffix removal. For example, if we want to stem the word زمان (in English “times”), we remove the first two characters زم and feed the remaining characters ان to the DFA. Readers can observe that the DFA will halt in state 3 and the driver returns this state and the word to the post processor. On the other hand, stemming the word پسران (in English “boys”), will return state 8 to the post processor.

The post processor uses this final state to determine a suffix group. SuffixGroup[] is a one-dimensional array. If  $F$  is the final state, SuffixGroup[ $F$ ] gives the identifier for the suffix group. This identifier is used to strip the suffixes. For example, in the case of our first word زمان, Suffix[3] returns NIL which means no suffix will be stripped from this word (the stem will be too short). In the case of the word پسران, the Suffix[8] returns PL2, which signifies the word is plural and two characters should be removed. hence the stemmer will return پسر (in English “boy”). Table 7 shows a suffix group array for the DFA in Figure 1.

If the identified suffix is possessive or plural, the post processor will feed the stripped word to the DFA to identify other suffixes. When the post processor identifies a verbal suffix state such as 10, it routes the stripped word for prefix identification to another DFA which is similar to our suffix DFA.

## 5 Evaluation

To evaluate the Farsi stemmer, we observed its effect on precision/recall using our Farsi information retrieval system (a vector-based system) [4], a fixed set of Farsi queries, and a fixed document collection.

A collection of 1647 Farsi documents, primarily Internet documents, was created. Native Farsi speakers compiled a list of sixty queries. For each document in the collection, and for each query, it was determined whether the document was relevant to the query. The Farsi collection was indexed without using the stemmer, and without removing stopwords [5]. We then processed each query in the list. The identical procedure was repeated except that the stemmer and the stopword list were introduced.

The 11-point average precisions are given in following tables. The test in which the stemmer was used shows an increase .033, or 18%.

	recall	precision	interpolated	
	0	0.328	0.483	
	10	0.253	0.353	
	20	0.228	0.273	
	30	0.119	0.215	
	40	0.131	0.197	
	50	0.151	0.170	
	60	0.078	0.105	
	70	0.055	0.074	
	80	0.039	0.060	
	90	0.027	0.049	
	100	0.046	0.046	
<hr/>				
	eleven pt avg	int eleven pt avg	three pt avg	int three pt avg
	0.132	0.184	0.139	0.167

Table 8: Average precision/recall results using no stemmer and no stopword removal

## 6 Conclusion and Further Research

The results of our stemming test indicate that the Farsi stemmer improves retrieval. Our tests were done on a small collection, so the effect of the stemmer on bigger collections is not known at this time. There are many ways that the stemmer can be modified. Possible modifications include editing the list of suffixes, changes to the minimum stem length, and foregoing prefix removal.

## References

- [1] A. Gharib, M. Bahar, B. Fooroozanfar, J. Homaii, and R. Yasami. *Farsi Grammar*. Jahane Danesh, 2nd edition, 2001.
- [2] David A. Hull. Stemming algorithms—a case study for detailed evaluation. Technical report, Rank Xerox Research Centre, Meylen, France, June 1995.
- [3] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

	recall	precision	interpolated
	0	0.342	0.544
	10	0.310	0.413
	20	0.290	0.333
	30	0.142	0.242
	40	0.137	0.210
	50	0.171	0.191
	60	0.096	0.141
	70	0.086	0.106
	80	0.045	0.080
	90	0.030	0.065
	100	0.060	0.060
<hr/>			
eleven pt avg	int eleven pt avg	three pt avg	int three pt avg
0.155	0.217	0.169	0.201

Table 9: Average precision/recall results using stemmer and stopword removal

- [4] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw Hill, New York, 1983.
- [5] Kazem Taghva, Russell Beckley, and Mohammad Sadeh. A list of farsi stopwords. Technical Report 2003-01, Information Science Research Institute, University of Nevada, Las Vegas, July 2003.
- [6] Kazem Taghva, Ron Young, Jeff Coombs, Ray Pereda, Russell Beckley, and Mohammad Sadeh. Farsi searching and display technologies. In *Proc. of the 2003 Symp. on Document Image Understanding Technology*, pages 41–46, Greenbelt, MD, April 2003.